UNIVERSITY OF CALIFORNIA, MERCED

OPTIMAL CONTROL WITH MODEL REDUCTION AND MACHINE LEARNING

by

Anni Zhao

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Mechanical Engineering

Committee in charge:
Professor Jian-Qiao Sun, Chair
Professor YangQuan Chen
Professor Reza Ehsani
Professor Roummel Marcia

The dissertation of Anni Zhao is approved:

Jian-Qiao Sun, Chair	Date
YangQuan Chen	Date
Reza Ehsani	Date
Roummel Marcia	Date

University of California, Merced

To my parents, Jiajiang Zhao and Lina An.

ACKNOWLEDGEMENTS

Research is a lifelong commitment for pursuing truth and wisdom. I am grateful for the opportunity to embark on this journey. First, I would like to express my greatful gratitude to my parents. No matter what happens, my parents always support and trust me with confidence. They give me solid backup and guide me to live a better life. They are the best parents to me in the world. I also want to express my gratitude to my advisor Dr. Sun, for his kindness, guidance, and patience. I am very fortunate to be one of his students. Dr. Sun always lead me with endless patience and help me to get through all the difficulties that I encountered during research. He taught me how to become a good researcher, a good student and a good person. I also want to express my gratitude to the former and current lab mates and colleagues, Dr. Shangjie Ma, Dr. Ghazaale Leylaz, Dr. Cheng Chen, Dr. Ala'aldin Alafaghani and Dr. Arash Toudeshki. They are all good colleagues to work with.

I would like to thank all of the committee members, Prof. Yangquan Chen, Prof. Reza Ehsani and Prof. Roummel Marcia. I am fortunate to have had the opportunity to enroll in Prof. Chen's Fractional Order Mechanics and Nonlinear Controls course. These classes help me broaden my eyesight and enrich my knowledge. Thank you Prof. Ehsani for helping me build the Delta robot, which broadened my research topic significantly. I am also want to say thank you to Prof. Marcia for being my committee.

At last, I want to express my gratitude to my husband Haoyu for his love, support, listening, and good temper. Specifically, I want to say thank you to my cat Luna. I have to admit that she is always (sleeping) by my side to help me get through all the stressful and painful moments without any complains.

I express my deep gratitude to all of them for shaping me into the person I am today. Without them, I can not be who I am today.

CURRICULUM VITAE

Education

B.S. in Mechatronics Engineering, Jimei University (Xiamen, China), 2016.

M.S. in Mechanical Engineering, Florida Institute of Techonology (Florida, USA), 2018.

M.S. in Mechatronics Engineering, Shijiazhuang Tiedao University (Shijiazhuang, China), 2019.

Honors

Earle C. Anthony Fellowship, University of California Merced (2020 and 2021).

Mechanical Engineering Bobcat Fellowship, University of California Merced (2021, 2022 and 2023).

Publications

A. Zhao, J.Q. Sun (2022): "Control for stability improvement of high-speed train bogie with a balanced truncation reduced order model". *Vehicle System Dyanmics*, 1-21.

A. Zhao, J. Huang, J.Q. Sun (2022): "Estimation of wheel–rail structural interactions from motion signals of high-speed train bogie". *Internationa Journal of Dynamics and Control*, 1-12.

A. Zhao, J.Q. Sun (2022): "Wear estimation of high speed train from motion measurements". *Journal of Vibration Testing and System Dynamics*, in press.

X. Wang, J. Jiang, L. Hong, A. Zhao, J.Q. Sun (2023): "Radial basis function neural networks solution for stationary probability density function of nonlinear stochastic systems". *Probabilistic Engineering Mechanics*, 71.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Different challenges exist in the area of control, including but not limited to the high computational cost, model uncertainty, disturbances, nonlinearity, multi-objective optimization, constraints handling of partial differential equations, implementation, and real-time computation. In this dissertation, we focused on solving the issues of high computational cost, model uncertainty, nonlinearity, and disturbances in control design. The curse of dimensionality is a common issue for control design. Many mechanical systems with multi-degrees of freedom are underactuated by design. The dynamics of these systems live in a relatively high dimensional state space. However, many states are less controllable or observable for the high dimensional system. To keep the most controllable and observable states in the system, it is highly advantageous to develop a reduced order model. It turns out to be highly advantageous to develop a reduced order model to keep the most controllable and observable states in the system. Moreover, the curse of dimensionality also exists in the computation of neural networks solutions. The high computational cost is a severe issue in solving the neural network solutions in optimal control. In this dissertation, we adopted the balanced truncation and empirical balanced truncation to reduce the order of the dynamics systems. Moreover, the radial basis function neural networks (RBFNN) are adopted to solve the Hamilton-Jacobi-Bellman (HJB) equation in optimal control. The high-speed train bogie and the Quanser Qube2 system are adopted as practical applications to test the performance of the reduced order model-based control design. Especially for the Qube2 system, extensive numerical simulations and experimental validations show that the RBFNN with a reduced order model save the computational power and improve the optimal controller's robustness. Furthermore, transfer learning is adopted to improve the performance of the radial basis function based HJB solutions with limited experimental data. Except for reducing the dimension of the system, a new neural networks structure - separable Gaussian neural networks is proposed to reduce the computation cost by making use of the factorizable property of the Gaussian activation functions in RBFNN. Numerical simulation results show that the separable gaussian functions reduce the computation cost and the required parameters in neural networks to solve the partial differential equations.

Model uncertainty and disturbances also play an important role in control design. The Luenberger, extended state observers and neural networks are adopted to estimate the unmeasurable system states and approximate unmodelled system dynamics. The minimum number of sensors to estimate the unknown dynamics are investigated with the help of the special structure of the extended state observer. With the unknown dynamics estimated, the recursive least squares algorithm is adopted to identify the parameters in the unknown dynamics if the persistent excitation condition is satisfied. Numerical examples show that the extended state observer performs well in estimating the unknown dynamics with a limited number of sensors. Except for using the observers, neural networks are also adopted to approximate the unknown dynamics of the system. Here, we adopted the Delta robot as an engineering application to illustrate the performance of a neural networks model with sliding mode control. By leveraging the machine learning techniques, the model uncertainty issue can be well-solved during the control design.

Chapter 1

INTRODUCTION

1.1 Background

1.1.1 Nonlinear Optimal Control

The theory of optimal control has been well developed and documented [3, 4]. The literature on applications of optimal controls is vast. Various optimal controls for linear systems such as the linear quadratic regulator (LQR) have been well studied and reported in the literature. However, when the dynamic system is nonlinear and when the performance is not quadratic, the solution of optimal controls is quite challenging to obtain. One popular way to formulate the optimal control problem is to make use of the HJB equation. There have been many studies of HJB equations in the mathematics and controls community.

The HJB equation is a nonlinear partial differential equation for the value function of the optimal control problem after the control in terms of the value function is substituted, and is subject to a terminal condition when the control horizon is finite. There have been many studies on the solution of the HJB equation. Crandall and Lions proposed the viscosity solution of the HJB equation [5], and found that the HJB equation exists a non-smoothness solution. A sequential alternating least squares method for solving a high dimensional linear HJB equation was proposed in [6]. In addition to the numerical methods such as viscosity solution, finite difference and finite element, neural networks are other promising approaches that have garnered recent interest [7].

Neural networks have been extensively studied for solving partial differential equations (PDEs) [8–10]. The well-known classical Ritz method was also implemented with the help of neural networks [11, 12]. A comprehensive survey on numerical algorithms, including the Monte Carlo algorithm and machine learning, for solving high dimensional PDEs can be found in [13]. It is in general difficult to obtain the analytical solutions of the HJB equation. Neural networks have also been applied to the HJB equation in the past thirty years [14, 15]. The value function is approximated by neural networks, which can be trained in the spatial domain or along the response trajectory. A nearly optimal solution for nonlinear systems in the spatial domain was obtained in the form of the neural networks with polynomial base functions [16]. The least squares algorithm with a time-domain discounting factor was introduced to improve the smoothness and convergence of neural networks training to find the optimal control in the spatial domain [17]. A neural-networks-based online HJB solution for uncertain nonlinear systems was developed in [18]. A RBF neural networks and a query network with warm-up network weights [19] were adopted to compute the HJB solution along response trajectory in [20]. The HJB equation with action constraints [16, 21] and fixed final time [22, 23] have also been studied with the help of neural networks. To incorporate the control limits in the cost function, a convex conjugate function was introduced to transform the constrained optimization problem into an unconstrained problem in [24]. The normalized Gaussian networks were adopted to develop a reinforcement learning framework to obtain the global value function for continuous-time systems while training the networks along different trajectories [25]. The actor-critic reinforcement learning together with the RBFNN was also applied to obtain the optimal control for a discrete-time system [20].

Recently, interest in solving the HJB solution in high dimensional state space has been increasing [26, 27]. A neural network with the sigmoid activation function is proposed to compute the viability sets for nonlinear continuous systems in [28]. A causality-free neural network method is proposed to solve high dimensional HJB equations that include the co-state vector as part of the solution [29,30]. Two neural networks architectures were introduced using minimum plus algebra to solve high dimensional optimal control problems with specified terminal costs [31]. Indeed, neural networks can obtain excellent approximations of the value function and the solution of the HJB equation leading to the optimal policy.

1.1.2 Challenges

Challenges exist in the area of optimal control, including but not limited to the high computational cost, model uncertainty, nonlinearity, multi-objective optimization, constraints handling of partial differential equations, practical implementation and real-time computation [32]. It is computationally intensive to solve complex optimization problems in optimal control for some cases, especially when the dimension of the system is relatively high. This issue can also be observed in the computation of neural networks to solve the HJB equation. The curse of dimensionality is a common issue in the neural networks solutions for solving the HJB equation. When the dimension $n \geq 4$, it is computational cost to solve the neural networks. Most optimal control cases are characterized by constraints on control inputs, state variables and solutions, resulting in boundary value problems in ordinary differential equations. Additionally, in real applications there exists nonlinearity and uncertain parameters in the model. Most optimal control algorithms are performed in simulation and may be not applicable in practice. As a result, solving optimal control problems can be quite challenging.

1.2 Contributions

As mentioned in Section 1.1.2, solving the optimal control problems is challenging. In this dissertation, we intended to address the issues of high-dimensionality, nonlinearity, model uncertainty, implementaion, and real-time computation in optimal control. Many mechanical systems with multi-degrees of freedom are underactuated by design. The dynamics of these systems live in a relatively high dimensional state space. Discovering a reduced order model of the system is highly advantageous, as it allows for limited control efforts to be focused on stabilizing the system and achieving improved tracking performance. Moreover, solving for the HJB equation for nonlinear control systems can be numerically complex, but a reduced order model can help to decrease these complexities. To ensure effective implementation of the control algorithm, it is crucial to find a reduced order model.

The model reduction technique - balanced truncation is applied to obtain the reduced order model of the system. To test its performance, the linear quadratic regulator is adopted as a special case. Several state observers, including the Luenberger observer and extended state observer are adopted to estimate the states and unknown dynamics in the systems since LQR is a full state feedback control. The observability of the extended state observer is analyzed to investigate the minimum number of sensors to estimate the system states.

The extensive use of neural networks has been observed in solving the HJB equation in optimal control problems. However, there are still many issues with the neural networks solution of the HJB equation, such as high computational cost of deep neural networks, generality of neural networks architecture, and requirements for extensive training data. Moreover, less investigation is done on the performance of optimal controls obtained in terms of the RBFNN. The RBFNN with Gaussian activation functions are adopted to solve the HJB equation for various optimal control problems and demonstrate the robustness of the optimal controls with the model reduction and transfer learning technique in machine learning.

In summary, the main contributions of this dissertation are addressed in the following:

1. To overcome the numerical issues of the high-dimensional system, we adopted the model reduction technique to reduce the order of the system. The optimal control solution from the reduced-order model was tested and proved to outperform the original model-based control in terms of robustness. Furthermore, the performance of the optimal policy is maintained while reducing the computational complexity of neural networks.

2. Instead of using a deep neural network, the RBFNN with one hidden layer is adopted to solve the HJB equations for multi-degree-of-freedom systems with high control performance in regions of the state space far away from equilibrium in trajectory tracking applications. The RBFNN solutions perform well

both in simulations and practicals. Furthermore, the RBFNN's single hidden layer property enhances its applicability by facilitating hardware implementation and real-time computation.

3. As the control determined by the HJB equation uses the nominal model of the physical system with uncertainties and disturbances, the transfer learning technique is adopted to further improve the performance of the optimal control with experimental data to compensate for the model uncertainties.

4. Once the system is effectively controlled, the extended state observer is adopted to estimate the unknown dyanmics. The minimum number of sensors required to estimate the unknowns is determined with the help of the special structure of the extended state observer and the Hautus test in linear control.

1.3 Dissertation Organization

The rest of the dissertation is organized as follows: In Chapter 2 "Nonlinear Optimal Control", the optimal control problem is formulated by discussing the Hamilton-Jacobi-Bellman equation and linear quadratic regulator. This chapter is the fundamental for understanding nonlinear optimal control problems.

In Chapter 3 "Model Reduction for Control Design", the balanced truncation and empirical balanced truncation are introduced to obtain the reduced order model. The reduced order model based optimal control is designed to stabilize the dynamic system. This chapter is the fundamental to understand the model reduction techniques and the implementation in control design.

In Chapter 4 "RBFNN Optimal Control with Model Reduction and Transfer Learning", the RBFNN is introduced to solve the optimal control problems. Furthermore, the model reduction and transfer learning techniques are introduced in this chapter to overcome the curse of dimensionality and to deal with model uncertainties.

In Chapter 5 "Extended State Observer with System Identification", the extended state observer is adopted to estimate the unknown dynamics of the system. The observability of the extended state observer is analyzed to investigate the minimum number of sensors to estimate the unknowns. The estimated dynamics can be further used for parameters identification with recursive least squares algorithm.

In Chapter 6 "Engineering Application I", the algorithms developed from Chapter 2 to Chapter 3 are applied to the high-speed train bogie.

In Chapter 7 "Engineering Application II", a neural network model of a Delta robot is developed and is used to design a sliding mode control for trajectory tracking applications.

Chapter 8 concludes the dissertation and briefly discusses future work for current algorithms. Most of the chapters in the dissertation are based upon the published papers [33–35]. Chapter 9 serves as the appendix.

Chapter 2

NONLINEAR OPTIMAL CONTROL

In this chapter, the optimal control problem for nonlinear systems is formulated. The HJB equation is explained in detail and the linear quadratic regulator is introduced as a special optimal control problem for linear systems.

2.1 Optimal Control Formulation

Consider a nonlinear dynamic system governed by the state equation

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}, t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.1}$$

where t_0 is the initial time, $\mathbf{x}_0$ is the initial state, $\mathbf{x} \in \mathbb{R}^{n\times 1}$, $\mathbf{u} \in \mathbb{R}^{m\times 1}$ and $\mathbf{F}(\cdot)$ is a nonlinear function of its arguments. Define a *target set* at a terminal time $t = T$ as

$$\Psi(\mathbf{x}(T), T) = 0, \tag{2.2}$$

where $\Psi \in \mathbb{R}^{p\times 1}$. Ψ defines a set where the terminal state $\mathbf{x}(T)$ must settle at time T.

The optimal control problem amounts to finding a control $\mathbf{u}$ in an admissible set $U \subset \mathbb{R}^{m\times 1}$ that drives the system from the initial state $\mathbf{x}_0$ to the target set $\Psi(\mathbf{x}(T), T)$ such that the following performance index J is minimized while the state equation is satisfied.

$$J = \phi(\mathbf{x}(T), T) + \nu^T \Psi(\mathbf{x}^T, T) + \int\limits_{t_0}^{T} \left[\mathcal{L}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) + \boldsymbol{\lambda}^T(t)(\mathbf{F}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}})\right] d\tau, \tag{2.3}$$

where $\phi(\mathbf{x}(T), T) \geq 0$ is the terminal cost, and $\mathcal{L}(\mathbf{x}(t), \mathbf{u}(t), t) > 0$ is called the *Lagrange function.*

The above optimal control statement defines a constrained optimization problem. With the help of the method of Lagrange multiplier, the following equations are obtained to determine the optimal control.
State Equation

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \boldsymbol{\lambda}} = \mathbf{F}(\mathbf{x}, \mathbf{u}, t). \tag{2.4}$$

Co-state Equation

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial \mathcal{L}}{\partial \mathbf{x}} - \frac{\partial \mathbf{F}^T}{\partial \mathbf{x}}\boldsymbol{\lambda}. \tag{2.5}$$

Optimality Condition

$$\frac{\partial H}{\partial \mathbf{u}} = 0. \tag{2.6}$$

Initial and Terminal Conditions

$$- H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t_0)dt_0 + \boldsymbol{\lambda}^T(t_0)d\mathbf{x}(t_0) = 0, \tag{2.7}$$

$$\begin{aligned} &\left[\frac{\partial}{\partial \mathbf{x}}\left(\phi + \Psi^T\nu\right) - \boldsymbol{\lambda}(T)\right]^T d\mathbf{x}(T) + \\ &\left[\frac{\partial}{\partial t}\left(\phi + \Psi^T\nu\right) + H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, T)\right] dT = 0. \end{aligned} \tag{2.8}$$

Target Set

$$\Psi\left(\mathbf{x}(T), T\right) = 0. \tag{2.9}$$

where the *Hamiltonian function* H is defined as

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) = \mathcal{L}(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T(t)\mathbf{F}(\mathbf{x}, \mathbf{u}, t). \tag{2.10}$$

When the admissible control $\mathbf{u}$ lies in a bounded subset U of $\mathbb{R}^{m\times 1}$, the optimality condition may not hold in U. In this case, it is replaced by *Pontryagin's minimum principle*, which is stated as

$$H(\mathbf{x}^*, \mathbf{u}^*, \boldsymbol{\lambda}^*, t) \leq H(\mathbf{x}^*, \mathbf{u}, \boldsymbol{\lambda}^*, t), \quad \forall \mathbf{u} \in U \subset \mathbb{R}^{m\times 1}, \tag{2.11}$$

where $\mathbf{x}^*$, $\mathbf{u}^*$ and $\boldsymbol{\lambda}^*$ denote the optimal solutions. Another way to state the minimum principle is

$$\mathbf{u}^* = \arg\left[\min_{\forall \mathbf{u}\in U\subset\mathbb{R}^{m\times 1}} H(\mathbf{x}^*, \mathbf{u}, \boldsymbol{\lambda}^*, t)\right]. \tag{2.12}$$

Note that Pontryagin's minimum principle is both a necessary and sufficient condition, while the optimality condition (2.6) is only necessary.

2.2 The Hamilton-Jacobi-Bellman Equation

Define a *value function* along the optimal trajectory based on the performance index as

$$\begin{aligned} V(\mathbf{x}^*(t),t) &= \phi(\mathbf{x}^*(T),T) + \int_t^T \mathcal{L}(\mathbf{x}^*(\tau),\mathbf{u}^*(\tau),\tau)d\tau \\ &= \min_{\mathbf{u}} \left[\phi(\mathbf{x}^*(T),T) + \int_t^T \mathcal{L}(\mathbf{x}^*,\mathbf{u},\tau)d\tau \right]. \end{aligned} \tag{2.13}$$

By definition, the following properties of the value function are given as

$$\begin{aligned} V(\mathbf{x}^*(T),T) &= \phi(\mathbf{x}^*(T),T), \\ V(\mathbf{x}^*(t),t_0) &= J. \end{aligned} \tag{2.14}$$

Take the derivative of the value function with respect to time t based on the definition in Equation (2.13)

$$\frac{dV(\mathbf{x}^*(t),t)}{dt} = -\mathcal{L}(\mathbf{x}^*(t),\mathbf{u}^*(t),t). \tag{2.15}$$

A remark on the role of the value function is in order. By definition,

$$V(\mathbf{x}^*(t),t) \geq 0, \quad \frac{dV(\mathbf{x}^*(t),t)}{dt} \leq 0. \tag{2.16}$$

Hence, $V(\mathbf{x}^*(t),t)$ is a Lyapunov function. The existence of the Lyapunov function implies that the system under the optimal control $\mathbf{u}^*(t)$ is stable.

By treating $V(\mathbf{x}^*(t),t)$ as a multi-variable function of the state and time, applying the chain rule of differentiation along the optimal path to obtain

$$\begin{aligned} \frac{dV(\mathbf{x}^*(t),t)}{dt} &= \frac{\partial V(\mathbf{x}^*(t),t)}{\partial t} + \frac{\partial V(\mathbf{x}^*(t),t)^T}{\partial \mathbf{x}}\dot{\mathbf{x}}^*(t) \\ &= \frac{\partial V(\mathbf{x}^*(t),t)}{\partial t} + \frac{\partial V(\mathbf{x}^*(t),t)^T}{\partial \mathbf{x}}\mathbf{F}(\mathbf{x}^*(t),\mathbf{u}^*(t),t). \end{aligned} \tag{2.17}$$

Equating these two derivatives yields

$$\begin{aligned}\frac{\partial V(\mathbf{x}^*(t),t)}{\partial t} &= -\mathcal{L}(\mathbf{x}^*(t),\mathbf{u}^*(t),t) \\ &- \frac{\partial V(\mathbf{x}^*(t),t)^T}{\partial \mathbf{x}}\mathbf{F}(\mathbf{x}^*(t),\mathbf{u}^*(t),t) \\ &= -H\left(\mathbf{x}^*(t),\mathbf{u}^*(t),\frac{\partial V(\mathbf{x}^*(t),t)}{\partial \mathbf{x}},t\right) \\ &= -\min_{\mathbf{u}} H\left(\mathbf{x}^*(t),\mathbf{u}(t),\frac{\partial V(\mathbf{x}^*(t),t)}{\partial \mathbf{x}},t\right).\end{aligned} \tag{2.18}$$

This is a partial differential equation defined in the state space governing the value function and is known as the *Hamilton-Jacobi-Bellman equation* (HJB).

According to the definition of the value function, we have a terminal condition for the HJB equation as $V(\mathbf{x}^*(T),T) = \phi(\mathbf{x}^*(T),T)$. When there are constraints imposed on the states, boundary conditions can also be imposed to the HJB equation.

The optimality condition in Equation (2.18) indicates that $\frac{\partial V}{\partial \mathbf{x}}$ plays the role of $\boldsymbol{\lambda}$ in the Hamiltonian in Equations (2.10) and (2.12). Hence, an expression of the optimal control in terms of the value function is obtained as,

$$\mathbf{u}^*(t) = -\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x}^*)\frac{\partial V(\mathbf{x}^*(t),t)}{\partial \mathbf{x}}. \tag{2.19}$$

2.3 Linear Quadratic Regulator

Consider the linear time invariant (LTI) system as a special case,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}(t), \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.20}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{2.21}$$

For the generality of discussions, we assume that $\mathbf{x} \in \mathbb{R}^{n\times 1}$, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{p\times n}$, $\mathbf{u} \in \mathbb{R}^{m\times 1}$, and $\mathbf{y} \in \mathbb{R}^{p\times 1}$. The optimal LQR control is designed to minimize the following performance index subject to the constraint of Equation (2.21) [36]

$$\mathcal{L}(\mathbf{x},\mathbf{u}) = \frac{1}{2}\left[\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)\right], \tag{2.22}$$

where the matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is semi-positive definite and $\mathbf{R} \in \mathbb{R}^{m\times m}$ is positive definite. The optimal control for this case is given by

$$\mathbf{u}^*(t) = -\mathbf{R}^{-1}\mathbf{B}^T(\mathbf{x}^*)\boldsymbol{\lambda}^*(t). \tag{2.23}$$

The optimality condition in Equation (2.18) indicates that $\frac{\partial V}{\partial \mathbf{x}}$ plays the role of $\boldsymbol{\lambda}$ in the Hamiltonian in Equations (2.10) and (2.12). Hence, an expression of the optimal control in terms of the value function is obtained as,

$$\mathbf{u}^*(t) = -\mathbf{R}^{-1}\mathbf{B}^T(\mathbf{x}^*)\frac{\partial V(\mathbf{x}^*(t), t)}{\partial \mathbf{x}}. \tag{2.24}$$

The LQR formulation for LTI system also leads to the Riccati equation

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = 0 \tag{2.25}$$

which yields the optimal gain $\mathbf{K}_{opt} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}$ such that the optimal feedback control reads

$$\mathbf{u}^*(t) = -\mathbf{K}_{opt}\mathbf{x}^* = -\mathbf{R}^{-1}\mathbf{B}^T(\mathbf{x}^*)\frac{\partial V(\mathbf{x}^*(t), t)}{\partial \mathbf{x}} \tag{2.26}$$

It can be shown that the algebraic Riccati equation (2.25) has a solution $\mathbf{P}$ such that $\mathbf{A} - \mathbf{B}\mathbf{P}^{-1}\mathbf{B}^T\mathbf{P}$ is a stable matrix. The optimal feedback law in Equation (2.26) minimizes the performance index in Equation (2.22).

Chapter 3

MODEL REDUCTION FOR CONTROL DESIGN

The balanced truncation model reduction (BTMR) method was first proposed by Moore in 1981 with the help of signal processing techniques–the principal component analysis and singular value decomposition [37]. Implementation of the method involves quite some matrix computations and proper definitions of the truncation error and its bounds. The BTMR method has since received much attention from the research community. An excellent review of computational methods and error bounds can be found in [38]. The BTMR method in the state-space representation for discrete-time systems is discussed in [39]. Model reduction in finite time and frequency intervals is examined in [40]. A simultaneous diagonalization algorithm is studied in [41]. The singular value decomposition for computing the balancing transformations is proposed, and proved to be a practical and popular calculation technique [41]. In this chapter, the balanced truncation algorithm is adopted to derive the reduced-order model of the system, followed by the development of control design based on the low-dimensional model.

3.1 Balanced Truncation

In this section, the BTMR method is introduced for control design. Recall the LTI system in Equation (2.21),

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}(t), \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{3.2}$$

However, the traditional BTMR method is only applicable to stable LTI systems. To deal with unstable LTI systems, we decompose the system into stable and unstable subsystems with the help of Schur decomposition and another transformation determined by the Sylvester equation. The balanced truncation order reduction is then applied to the stable subsystem. A global transformation for the whole system is then defined that combines several matrices from Schur decomposition and balanced truncation. We first discuss the method of Schur decomposition.

3.1.1 Schur Decomposition

The real Schur transformation can decompose the system into stable and unstable subsystems [42]. Two steps are included leading to the stable-unstable decomposition of the system without the coupling between two subsystems.

The first step applies the real Schur decomposition to transform the system into an upper triangular block structure as shown below. For the system in (6.2), there exists an orthogonal matrix $\mathbf{U}$ such that $\mathbf{x} = \mathbf{U}\mathbf{x}_{rs}$. The transformed system matrices are given as

$$\begin{aligned} \mathbf{A}_{rs} &= \mathbf{U}^{-1}\mathbf{A}\mathbf{U} = \begin{bmatrix} \mathbf{A}_{rs1} & \mathbf{A}_{rs2} \\ \mathbf{0} & \mathbf{A}_{rs3} \end{bmatrix} \\ \mathbf{B}_{rs} &= \mathbf{U}^{-1}\mathbf{B} = \begin{bmatrix} \mathbf{B}_{rs1} \\ \mathbf{B}_{rs2} \end{bmatrix} \\ \mathbf{C}_{rs} &= \mathbf{C}\mathbf{U} = \begin{bmatrix} \mathbf{C}_{rs1} & \mathbf{C}_{rs2} \end{bmatrix} \end{aligned} \tag{3.3}$$

where $\mathbf{A}_{rs1} \in \mathbb{R}^{n_{us}\times n_{us}}$, $\mathbf{A}_{rs2} \in \mathbb{R}^{n_s\times n_{us}}$, $\mathbf{A}_{rs3} \in \mathbb{R}^{n_s\times n_s}$, $n_s + n_{us} = n$, n_s is the dimension of the stable subsystem and n_{us} is the dimension of the unstable subsystem. $\mathbf{x}_{rs}$ denotes the states after the real Schur decomposition, $\mathbf{A}_{rs3}$ corresponds to the stable subsystem, $\mathbf{A}_{rs1}$ represents the unstable subsystem. The upper right block $\mathbf{A}_{rs2}$ describes the coupling between the stable and unstable subsystems.

In the second step, we attempt to decouple the stable and unstable subsystems by considering a transformation $\mathbf{S}$ determined by the Sylvester equation,

$$\mathbf{A}_{rs1}\mathbf{S} - \mathbf{S}\mathbf{A}_{rs3} = \mathbf{A}_{rs2} \tag{3.4}$$

A unique solution $\mathbf{S} \in \mathbb{R}^{n_{us}\times n_s}$ of the Sylvester equation exists when the matrices $\mathbf{A}_{rs1}$ and $\mathbf{A}_{rs3}$ do not have common eigenvalues [43]. A new transformation $\mathbf{V}$ for the full state vector that can fully decouple the stable and unstable subsystems is given by [44]

$$\mathbf{V} = \begin{bmatrix} \mathbf{I}_{us} & -\mathbf{S} \\ \mathbf{0} & \mathbf{I}_s \end{bmatrix} \tag{3.5}$$

where $\mathbf{I}_{us} \in \mathbb{R}^{n_{us}\times n_{us}}$ and $\mathbf{I}_s \in \mathbb{R}^{n_s\times n_s}$ are both identity matrices. It is interesting to show that the inverse of $\mathbf{V}$ can be obtained as

$$\mathbf{V}^{-1} = \begin{bmatrix} \mathbf{I}_{us} & -\mathbf{S} \\ 0 & \mathbf{I}_s \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I}_{us} & \mathbf{S} \\ 0 & \mathbf{I}_s \end{bmatrix} \tag{3.6}$$

Define the transformation $\mathbf{x}_{rs} = \mathbf{V}\mathbf{x}_d$. $\mathbf{x}_d$ is the new state vector of the transformed system whose stable and unstable states are fully decoupled, as shown next. The state vector $\mathbf{x}_d = [\mathbf{x}_{us}, \mathbf{x}_s]^T$ satisfies the following equation.

$$\begin{aligned} \dot{\mathbf{x}}_d(t) &= \mathbf{A}_d\mathbf{x}_d(t) + \mathbf{B}_d\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}_d\mathbf{x}_d(t) \end{aligned} \tag{3.7}$$

where $\mathbf{x}_s \in \mathbb{R}^{n_s \times 1}$ and $\mathbf{x}_{us} \in \mathbb{R}^{n_{us} \times 1}$,

$$\mathbf{A}_d = \mathbf{V}^{-1}\mathbf{U}^{-1}\mathbf{A}\mathbf{U}\mathbf{V} = \begin{bmatrix} \mathbf{A}_{us} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_s \end{bmatrix} \tag{3.8}$$

$$\begin{aligned} \mathbf{B}_d &= \mathbf{V}^{-1}\mathbf{U}^{-1}\mathbf{B} = \begin{bmatrix} \mathbf{B}_{us} \\ \mathbf{B}_s \end{bmatrix} \\ \mathbf{C}_d &= \mathbf{C}\mathbf{U}\mathbf{V} = \begin{bmatrix} \mathbf{C}_{us} & \mathbf{C}_s \end{bmatrix} \end{aligned} \tag{3.9}$$

$\mathbf{A}_{us}$ denotes the state matrix of the unstable part of the system. $\mathbf{A}_s$ denotes the stable part of the system. $\mathbf{B}$ and $\mathbf{C}$ are also decomposed into the stable and unstable parts. Clearly, the stable and unstable subsystems are decoupled.

In the following, we shall apply the BTMR method to develop a reduced order model of the stable subsystem. In other words, we shall work with the matrices $\mathbf{A}_s$, $\mathbf{B}_s$ and $\mathbf{C}_s$ of the stable subsystem.

3.1.2 Model-based Balanced Truncation

In this section, we present the procedure of the BTMR method in [41]. The balanced truncation reduced order model is realized by introducing a transformation $\mathbf{T}_s$ to the state $\mathbf{x}_s(t)$ such that $\mathbf{x}_s = \mathbf{T}_s\mathbf{x}_b$ which leads to a so-called balanced form of the system. In the following, we show how to find the transformation $\mathbf{T}_s$ in order to obtain the desired balanced order system.

Let $\mathbf{W}_c$ and $\mathbf{W}_o$ be the controllability and observability Gramians of the stable subsystem. Consider two continuous time Lyapunov equations,

$$\begin{aligned} \mathbf{A}_s\mathbf{W}_c + \mathbf{W}_c\mathbf{A}_s^T + \mathbf{B}_s\mathbf{B}_s^T &= 0 \\ \mathbf{A}_s^T\mathbf{W}_o + \mathbf{W}_o\mathbf{A}_s + \mathbf{C}_s^T\mathbf{C}_s &= 0 \end{aligned} \tag{3.10}$$

If $\mathbf{A}_s$ is stable, the Lyapunov equations have unique symmetric positive definite solutions [45] $\mathbf{W}_c$ and $\mathbf{W}_o \in \mathbb{R}^{n_s \times n_s}$, which are the controllability and observability Gramians of the system [40, 46]. After obtaining $\mathbf{W}_c$ and $\mathbf{W}_o$, we compute the transformation $\mathbf{T}_s$ through the following steps.

1. Evaluate the Cholesky decomposition of Gramians $\mathbf{W}_c$ and $\mathbf{W}_o$,

$$\mathbf{W}_c = \mathbf{R}\mathbf{R}^T, \quad \mathbf{W}_o = \mathbf{Q}\mathbf{Q}^T \tag{3.11}$$

2. Evaluate the singular value decomposition of the Hankel matrix defined as $\mathbf{H} = \mathbf{Q}^T\mathbf{R}$,
$$\mathbf{H} = \mathbf{U}_H\mathbf{\Sigma}\mathbf{V}_H^T \tag{3.12}$$

3. The transformation $\mathbf{T}_s$ can be obtained as,
$$\mathbf{T}_s = \mathbf{R}\mathbf{V}_H\mathbf{\Sigma}^{1/2}, \quad \mathbf{T}_s^{-1} = \mathbf{\Sigma}^{-1/2}\mathbf{U}_H^T\mathbf{Q}^T \tag{3.13}$$
where $\mathbf{\Sigma} = diag(\sigma_1, \sigma_2, ..., \sigma_{n_s})$, $\sigma_1 > \sigma_2 > \cdots \sigma_{n_s} \geq 0$. σ_i are called the Hankel singular values of the matrix $\mathbf{H}$.

Define the transformed Gramians as $\bar{\mathbf{W}}_c = \mathbf{T}_s^{-1}\mathbf{W}_c\mathbf{T}_s^{-T}$ and $\bar{\mathbf{W}}_o = \mathbf{T}_s^T\mathbf{W}_o\mathbf{T}_s = \mathbf{\Sigma}$. The singular values $\mathbf{\Sigma}$ can be rewritten as
$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & 0 \\ 0 & \mathbf{\Sigma}_2 \end{bmatrix} \tag{3.14}$$
where $\mathbf{\Sigma}_1 = diag(\sigma_1, \sigma_2, ..., \sigma_r)$, $\mathbf{\Sigma}_2 = diag(\sigma_{r+1}, \sigma_{r+2}, ..., \sigma_{n_s})$. $\mathbf{\Sigma}_2$ will be truncated to obtain the reduced order system. Hence, in the transformed state $\mathbf{x}_b = \mathbf{T}_s^{-1}\mathbf{x}_s$, we neglect the components $x_{b,i}$ for $i = r+1, r+2, \cdots, n_s$. $r < n_s$ denotes the number of states to be included in the reduced order model of the stable subsystem.

Let $\mathbf{x}_r(t) \in \mathbb{R}^{r\times 1}$ denote the truncated state vector, $\mathbf{A}_r \in \mathbb{R}^{r\times r}$ be the corresponding state matrix, $\mathbf{B}_r \in \mathbb{R}^{r\times m}$ the reduced order control influence matrix and $\mathbf{C}_r \in \mathbb{R}^{p\times r}$ the reduced order output matrix. The reduced order system output $\mathbf{y}_r \in \mathbb{R}^{p\times 1}$ has the same dimension as that of the original output of the system. It will be shown later in Equation (3.26) that $\mathbf{y}_r$ is approximately equal to the original output $\mathbf{y}$. The reduced order system is given as,
$$\begin{aligned} \dot{\mathbf{x}}_r(t) &= \mathbf{A}_r\mathbf{x}_r(t) + \mathbf{B}_r\mathbf{u}(t) \\ \mathbf{y}_r(t) &= \mathbf{C}_r\mathbf{x}_r(t) \end{aligned} \tag{3.15}$$
where $\mathbf{A}_r = \mathbf{T}_{sr}^{-1}\mathbf{A}\mathbf{T}_{sr}$, $\mathbf{B}_r = \mathbf{T}_{sr}^{-1}\mathbf{B}$ and $\mathbf{C}_r = \mathbf{C}\mathbf{T}_{sr}$. $\mathbf{T}_{sr} \in \mathbb{R}^{n_s\times r}$ is part of the transformation $\mathbf{T}_s$ after the columns corresponding to the truncated states are removed. Let $\mathbf{x}_t \in \mathbb{R}^{(n_s-r)\times 1}$ represent the truncated states in $\mathbf{x}_b$. We rewrite $\mathbf{x}_b$ and the transformation $\mathbf{T}_s$ as
$$\mathbf{x}_b = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_t \end{bmatrix}, \quad \mathbf{T}_s = [\mathbf{T}_{sr}, \mathbf{T}_{st}], \tag{3.16}$$
where $\mathbf{T}_{st} \in \mathbb{R}^{n_s\times(n_s-r)}$. Hence, we have
$$\mathbf{x}_s = \mathbf{T}_s\mathbf{x}_b = \mathbf{T}_{sr}\mathbf{x}_r + \mathbf{T}_{st}\mathbf{x}_t. \tag{3.17}$$

Remark 3.1.1. The Gramian $\bar{\mathbf{W}}_c$ indicates the ability to control the states with the current control configuration and $\bar{\mathbf{W}}_o$ suggests the potential to observe the state from the output. The transformation $\mathbf{T}_s$ aims to make the truncated system equally controllable and observable such that the transformed Gramians are equal $\bar{\mathbf{W}}_c = \bar{\mathbf{W}}_o$, and share the same diagonal elements σ_i. This is the meaning of the word "balanced" in the name of the method. From energy point of view, the minimum energy to control a system to track the desired trajectories is measured by the integral of the control norm squared, as σ_i^{-1}, then smaller σ_i requires more energy to control and suggests that the corresponding states are less important [47].

We now return to the system with both the stable and unstable systems in Equation (3.7). Let $\mathbf{I}_{us} \in \mathbb{R}^{n_{us} \times n_{us}}$ be an identity matrix. The matrices for the balanced system can be obtained as,

$$\begin{aligned}\mathbf{A}_{BT} &= \begin{bmatrix}\mathbf{I}_{us} & 0\\ 0 & \mathbf{T}_s\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{A}_{us} & 0\\ 0 & \mathbf{A}_s\end{bmatrix}\begin{bmatrix}\mathbf{I}_{us} & 0\\ 0 & \mathbf{T}_s\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{I}_{us}^{-1} & 0\\ 0 & \mathbf{T}_s^{-1}\end{bmatrix}\mathbf{V}^{-1}\mathbf{U}^{-1}\mathbf{A}\mathbf{U}\mathbf{V}\begin{bmatrix}\mathbf{I}_{us} & 0\\ 0 & \mathbf{T}_s\end{bmatrix}\end{aligned} \tag{3.18}$$

Introduce a matrix $\mathbf{T}$

$$\mathbf{T} = \mathbf{U}\mathbf{V}\mathbf{T}_n, \quad \mathbf{T}_n = \begin{bmatrix}\mathbf{I}_{us} & 0\\ 0 & \mathbf{T}_s\end{bmatrix} \tag{3.19}$$

such that the transformed state matrix is given by

$$\begin{aligned}\mathbf{A}_{BT} &= \mathbf{T}_n^{-1}\mathbf{V}^{-1}\mathbf{U}^{-1}\mathbf{A}\mathbf{U}\mathbf{V}\mathbf{T}_n \\ &= \mathbf{T}^{-1}\mathbf{A}\mathbf{T}\end{aligned} \tag{3.20}$$

where $\mathbf{A}_{BT}$ denotes the balanced form of the matrix $\mathbf{A}$ of the original system. The same transformations can be applied to other matrices of the original system. Furthermore, we have $\mathbf{x} = \mathbf{T}\mathbf{x}_{BT}$ where $\mathbf{x}_{BT}$ is the full state vector of the transformed model and is defined as

$$\mathbf{x}_{BT} = \begin{bmatrix}\mathbf{x}_r\\ \mathbf{x}_t\end{bmatrix}. \tag{3.21}$$

Here, we must point out that $\mathbf{x}_r$ as part of $\mathbf{x}_{BT}$ at this point on may contain unstable modes if the original system is unstable.

Remark 3.1.2. Note that the Hankel singular values for the unstable subsystem do not exist and are infinite for the corresponding unstable dynamics of the system. The Hankel singular values for the stable subsystem are given by Equation (3.14). The order truncation will be applied to the whole balanced system to obtain the reduced order system. Hence, we obtain a reduced order system for LTI systems with both stable and unstable dynamics.

The accuracy of the response computed from the reduced order system has been well studied in the literature. An error bound can be defined to describe the quality of the approximation in frequency domain [48]. Consider the transfer functions from the input to the output of the original system and the reduced order system,

$$\begin{aligned} \mathbf{G}(s) &= \mathbf{C}_s(s\mathbf{I} - \mathbf{A}_s)^{-1}\mathbf{B}_s + \mathbf{D}_s \\ \hat{\mathbf{G}}(s) &= \mathbf{C}_r(s\mathbf{I} - \mathbf{A}_r)^{-1}\mathbf{B}_r + \mathbf{D}_r \end{aligned} \tag{3.22}$$

The error bound for the system with zero initial conditions can be expressed as

$$E_\infty = ||\mathbf{G}(s) - \hat{\mathbf{G}}(s)||_\infty \leq 2\gamma \tag{3.23}$$

where $||\cdot||_\infty$ denotes the H_∞ norm, $\gamma = \sigma_{r+1} + \cdots + \sigma_{n_s}$ and σ_i $(i = r+1, r+2, \cdots, n_s)$ are the truncated Hankel singular values in $\mathbf{\Sigma}_2$. A flowchart illustrating the steps of the balanced truncation model reduction method is shown in Figure 3.1.

Remark 3.1.3. When the original system is stable, there is no need for Schur decomposition and the subsequent transformation. The balancing transformation $\mathbf{T}_s$ can be applied to the original system directly.

3.1.3 Non-zero Initial Conditions

The error bound in Equation (3.23) is defined with the help of transfer functions in the frequency domain, which does not include the effect of non-zero initial conditions. When the balanced truncation model reduction method is applied to the system with non-zero initial conditions, the output error of the reduced order model can grow unbounded. There have been attempts to address the issue of [49–52]. In this section, the $\mathbf{X}_0$-balanced truncation method in [50] is adopted for the LTI system with non-zero initial conditions.

Assume that the original system (6.2) has an initial condition $\mathbf{x}_0$. Let $\mathbf{X}_0$ denote a matrix whose columns spanning a linear space, to which $\mathbf{x}_0$ belongs. Hence, there exists a vector $\mathbf{v}_0$ in the linear space such that $\mathbf{x}_0 = \mathbf{X}_0\mathbf{v}_0$. The simplest choice of $\mathbf{X}_0$ is such that each of its column is a unit base vector of the linear space, i.e. $\mathbf{X}_0$ is an identity matrix. That is, the spanned space is $\mathbb{R}^n$. For convenience, we shall denote the linear space spanned by the columns of $\mathbf{X}_0$ as $\mathbf{X}_0$. With this, we consider an augmented system defined as,

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + [\mathbf{B}, \mathbf{X}_0] \begin{bmatrix} \mathbf{u}(t) \\ \mathbf{u}_0(t) \end{bmatrix} \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{3.24}$$

where $\mathbf{u}_0(t)$ is an auxiliary input to generate the effect of non-zero initial conditions. The procedure of the BTMR method discussed earlier can be applied to Equation

(3.24) when the original system has an arbitrary initial condition $\mathbf{x}_0 \in \mathbf{X}_0$. This leads to the transformation $\mathbf{T}_s$, which is partitioned in Equation (3.16) to obtain the reduced order vector $\mathbf{x}_r$ in Equation (3.17). The reduced order system reads,

$$\dot{\mathbf{x}}_r(t) = \mathbf{A}_r \mathbf{x}_r(t) + [\mathbf{B}_r, \mathbf{X}_{0r}] \begin{bmatrix} \mathbf{u}(t) \\ \mathbf{u}_0(t) \end{bmatrix} \tag{3.25}$$
$$\mathbf{y}_r(t) = \mathbf{C}_r \mathbf{x}_r(t)$$

where $\mathbf{X}_{0r} = \mathbf{T}_{sr}^{-1} \mathbf{X}_0$.

The output error bound between the reduced order model (3.25) and the original system (3.24) reads,

$$\|\mathbf{y}(t) - \mathbf{y}_r(t)\|_{L_2} \leq 2\gamma \|\mathbf{u}(t)\|_{L_2} + 3\|\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{A}\|_2^{\frac{1}{3}} \|\mathbf{X}_0\|_2^{\frac{1}{3}} \gamma^{\frac{2}{3}} \|\mathbf{v}_0\|_2 \tag{3.26}$$

where $\|\cdot\|$ is the Euclidian norm of a vector. Detail discussions of the error bound analysis can be found in [50]. We should point out that Equation (3.25) for the reduced order model has no specified initial conditions, while Equation (6.2) has non-zero initial conditions. As discussed in [50], an impulsive function for $\mathbf{u}_0(t)$ can be selected to generate the equivalent effect of the same initial conditions as those for Equation (6.2).

Remark 3.1.4. Application of the BTMR method to the system (3.24) creates the transformations which in the end leads to the global transformation $\mathbf{T}$ defined in Equation (3.19). We then apply the global transformation $\mathbf{T}$ to the system with non-zero initial conditions defined in Equation (6.2), leading to the BT reduced order model of the system. When we simulate the BT reduced order model, we can now impose the same initial conditions as the one for the original system.

3.2 Empirical Balanced Truncation

The controllability and observability gramians used in the balanced truncation method are defined for linear systems and are usually model-based. For nonlinear systems without a detailed mathematical model, the controllability and observability gramians can be estimated from input-output data. These are called the empirical gramians. The empirical gramians yield the exactly same balanced truncation transformations for stable linear system. However, it has been studied that the reduced order model from the empirical gramians contains more accurate

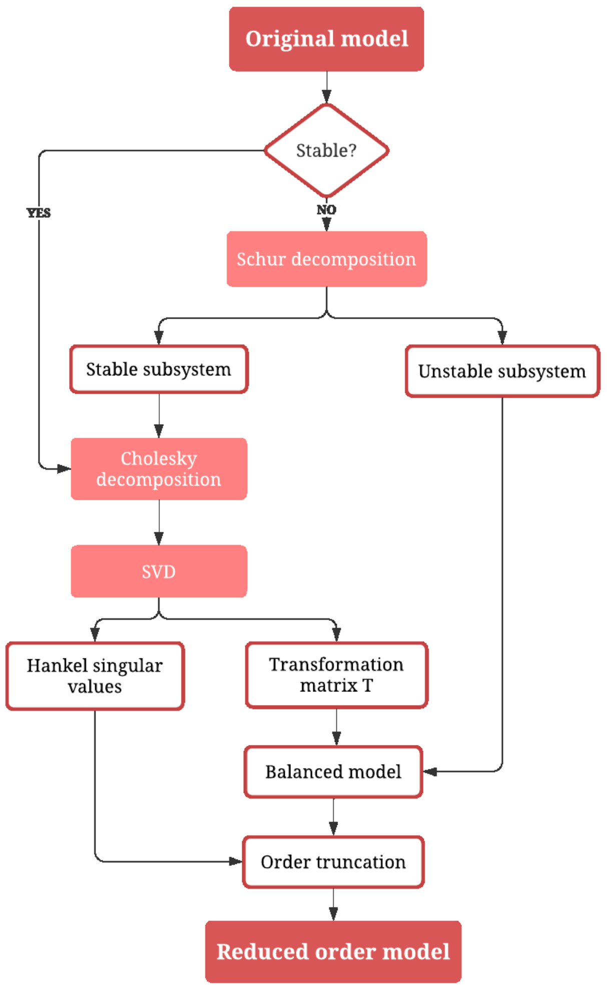


Figure 3.1: Flowchart of balanced truncation model reduction algorithm for LTI systems with unstable dynamics.

information than the traditional model-based balanced truncation [53,54]. The empirical controllability gramian and observability gramian are defined as,

$$\hat{\mathbf{W}}_c := \sum_{i=1}^{p}\sum_{m=1}^{s}\sum_{l=1}^{r}\frac{1}{rsc_m^2}\int_0^\infty \mathbf{\Phi}^{ilm}(t)dt \tag{3.27}$$

$$\hat{\mathbf{W}}_o := \sum_{m=1}^{s}\sum_{l=1}^{r}\frac{1}{rsc_m^2}\int_0^\infty \mathbf{T}_l\boldsymbol{\psi}^{lm}(t)\mathbf{T}_l^T dt \tag{3.28}$$

where

$$\tau^n = \{\mathbf{T}_1, \cdots, \mathbf{T}_r; \mathbf{T}_i \in \mathbb{R}^{n\times n}, \mathbf{T}_i^T * \mathbf{T}_i = \mathbf{I}, i = 1, \cdots, r\} \tag{3.29}$$

$$M = \{c_1, \cdots, c_s; c_i \in \mathbb{R}, c_i > 0, i = 1, \cdots, s\} \tag{3.30}$$

$$\varepsilon^n = \{\mathbf{e}_1, \cdots, \mathbf{e}_n\} \tag{3.31}$$

$$\mathbf{\Phi}^{ilm}(t) = (\mathbf{x}^{ilm}(t) - \bar{\mathbf{x}}^{ilm})(\mathbf{x}^{ilm}(t) - \bar{\mathbf{x}}^{ilm})^T \tag{3.32}$$

$$\boldsymbol{\psi}_{ij}^{lm}(t) = (\mathbf{y}^{ilm}(t) - \bar{\mathbf{y}}^{ilm})^T(\mathbf{y}^{jlm}(t) - \bar{\mathbf{y}}^{jlm}), \tag{3.33}$$

and $\mathbf{x}^{ilm}(t)$ is the state trajectories with impluse input given as $u(t) = c_m\mathbf{T}_l\mathbf{e}_i\delta(t)$, $\mathbf{y}^{ilm}(t)$ is the output trajectories of the system with the inital condition as $\mathbf{x}_0 = c_m\mathbf{T}_l\mathbf{e}_i$, $\mathbf{e}_i$ are unit vectors. Same calculation procedures from Equation (3.11) to Equation (3.13) can be applied to the empirical gramians to obtain the transformation matrix $\mathbf{T}$ and the reduced order model in Equation (3.15). It has been extensively investigated that the empirical gramians are applicable to obtain the reduced order model for the nonlinear systems using the empirical Gramian framework [55]. The key idea of the empirical gramians in Equation (3.28) is the averaging over local Gramians obtained from varying initial states around an operating point near a steady-state [56]. However, the reduced nonlinear model is locally since it is computed through data on different trajectories with certain initial conditions near a steady state. In this study, the empirical gramians are only applied to linear systems to make them compatible with the RBFNN optimal control algorithm mentioned in chapter 4 since RBFNN is designed to find the solution in a particular spatial domain. The result from empirical gramians could be misleading if RBFNN is computated over the region of the operation for the reduced order nonlinear system.

3.3 Reduced Order Model Control

We are now ready to design the control for dynamic system by using the reduced order model in Equation (3.15). The control is ultimately implemented on the original system. There are many options for the design of control. In this section, we use the LQR control to demonstrate the utility of the reduced order model. In connection with the control, a Luenberger observer is designed for the state estimation.

3.3.1 LQR Control

Recall the reduced order system (3.15). The optimal LQR control is designed to minimize the following performance index subject to the constraint of Equation (3.15) [36]

$$J = \frac{1}{2}\int_0^\infty [\mathbf{x}_r(t)^T\mathbf{Q}\mathbf{x}_r(t) + \mathbf{u}(t)^T\mathbf{R}\mathbf{u}(t)]dt \tag{3.34}$$

where the matrix $\mathbf{Q} \in \mathbb{R}^{r\times r}$ is semi-positive definite and $\mathbf{R} \in \mathbb{R}^{m\times m}$ is positive definite.

The optimal feedback control reads

$$\mathbf{u} = -\mathbf{K}_{opt}\mathbf{x}_r \tag{3.35}$$

In order to implement the feedback control in Equation (3.35) with the output $\mathbf{y}(t)$ of the original system, we must estimate the state vector $\mathbf{x}_r(t)$. Different type of observers, including the Luenberger observer, Kalman filter or extended state observer can be considered to estimate the state vector $\mathbf{x}(t)$. To this end, we adopt the well-documented Luenberger observer.

3.3.2 Luenberger Observer

For practical reasons, it is often not possible to measure all the states of a system. An observer can be designed to estimate the unmeasurable states. We decide to adopt the popular Luenberger observer. The structure of the Luenberger observer is shown below,

$$\dot{\hat{\mathbf{x}}}_r(t) = \mathbf{A}_r\hat{\mathbf{x}}_r(t) + \mathbf{B}_r\mathbf{u}(t) + \mathbf{L}[\mathbf{y}(t) - \hat{\mathbf{y}}_r(t)] \tag{3.36}$$

$$\hat{\mathbf{y}}_r(t) = \mathbf{C}_r\hat{\mathbf{x}}_r(t) \tag{3.37}$$

where $\mathbf{y}(t)$ is the original system output and $\hat{\mathbf{y}}_r(t)$ is an estimate of the output of the BT reduced order model.

According to Equation (3.26), $\mathbf{y}_r(t) \approx \mathbf{y}(t)$. In the following analysis, we replace $\mathbf{y}(t)$ with $\mathbf{y}_r(t)$. In the control implementation, the original system output $\mathbf{y}(t)$ is still used as the input to the Luenberger observer.

The output estimation error is then defined as $\mathbf{y}_r(t)-\hat{\mathbf{y}}_r(t)$. It is the feedback signal to the observer. $\mathbf{L}$ is the observer gain matrix. Define the state error as $\tilde{\mathbf{x}}_r(t) = \mathbf{x}_r(t) - \hat{\mathbf{x}}_r(t)$ where $\hat{\mathbf{x}}_r(t)$ is the estimate of the state $\mathbf{x}_r(t)$. From Equation (3.36), we obtain the equation governing the state estimation error,

$$\dot{\tilde{\mathbf{x}}}_r(t) = (\mathbf{A}_r - \mathbf{L}\mathbf{C}_r)\tilde{\mathbf{x}}_r(t) \tag{3.38}$$

We can show that the pair $(\mathbf{A}_r, \mathbf{C}_r)$ of the reduced order model is observable. Hence, we can find a gain matrix $\mathbf{L}$ such that $\mathbf{A}_r - \mathbf{L}\mathbf{C}_r$ is stable. The state estimation error will converge to zero.

3.4 Conclusions

We have presented a balanced truncation reduced order model for the high-dimensional dynamic systems, which can be severely underactuated and difficult to control. The balanced truncation model reduction is usually applied to stable systems. In this chapter, we first apply the real Schur decomposition to separate the stable and unstable modes of the dynamics system. The balanced truncation model reduction is then carried out for the stable subsystem. An augmented $\mathbf{X}_0$-balanced truncation algorithm is also considered to deal with the dependence of the balanced truncation model reduction on initial conditions. The reduced order subsystem is then combined with the unstable subsystem. A linear quadratic regulator control together with a Luenberger observer for the state estimation is designed for the dynamic system with non-zero initial conditions.

Chapter 4

RBFNN OPTIMAL CONTROL WITH MODEL REDUCTION AND TRANSFER LEARNING

While the solutions of HJB equation have been extensively studied, less research is done on the performance of optimal controls obtained in terms of the RBFNN. This chapter presents an approach based on radial basis function neural networks with Gaussian activation functions to solve the HJB equation for various optimal control problems and demonstrate the robustness of the optimal controls with the model reduction and transfer learning technique in machine learning. We address the robustness improvement of the optimal controls obtained with the RBFNN, model reduction and transfer learning.

Since the mathematical model of nonlinear control systems can never perfectly capture the dynamics of the real system, the RBFNN based control built with the mathematical model may need fine tuning or retraining when it is applied to the physical system in an experimental setting. This is where the transfer learning technique of machine learning comes in. After we obtain certain experimental data, we can partially retrain the neural networks in order to further improve the performance of the control.

4.1 Universal Approximation Theorem for Neural Networks

The universal approximation theorem [57] for neural networks states that a feedforward neural network with a single hidden layer can approximate any continuous function to arbitrary precision, given enough neurons in the hidden layer. This theorem demonstrates the capability of neural networks as a universal function approximator, meaning that the neural networks can be used to model various complex mathematical functions. There are different ways to formulate the neural networks sturctures, including the feedforward neural networks, convolutional neural networks, recurrent neural networks, autoencoders, generative adversarial networks and transformers. Among all the structures of neural networks, RBFNN is an example of feedforward neural networks with a single hidden layer and radial basis activation functions. The RBFNN satisfies the universal approximation property as shown in [58]. In this chapter, we adopted the RBFNN to approximate the value function in optimal control problems.

There are many studies on the method of RBFNN itself. An adaptive extended Kalman filter (AEKF) is developed for estimating the weights, centers, and width of RBFNN [59]. It has been shown that the RBFNN with one hidden layer and the same smoothing factor in each kernel are broad enough for universal approximation as concluded in [58]. An excellent review on RBFNN for regression and classification can be found in [60]. The advantages of RBFNN are compared with other neural networks architectures in [61]. An application of the RBFNN in control design can be found in [62] to approximate the error model in vibration suppression. It has been found that RBFNN have an advantage over the conventional sigmoid neural networks (NN) owing to that the nth-dimensional gaussian function is well-established from probability theory, Kalman filtering, *etc* [63].

4.2 Solution of Optimal Control Problems with Radial Basis Function Neural Networks

4.2.1 Neural Networks Solution of Value Function

Recall the optimal control problem with Hamilton-Jacobi-Bellman equation in Section 2.2. In the following, we shall focus on the special time-invariant case when the terminal time $T \to \infty$, the terminal cost $\phi(\mathbf{x}(T), T) \to 0$, and the value function is a function of the state only, i.e. $V = V(\mathbf{x})$.

We express the value function in terms of radial basis function neural networks with Gaussian activation functions,

$$\begin{aligned} V(\mathbf{x}, \mathbf{w}(k)) &= \sum_{j=1}^{N_G} w_j(k) g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) + C \\ &= \mathbf{w}^T(k)\mathbf{h}(\mathbf{x}) + C \\ &= \mathbf{h}^T(\mathbf{x})\mathbf{w}(k) + C, \end{aligned} \tag{4.1}$$

where N_G is the number of Gaussian functions $g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ and C is an arbitrary integration constant to guarantee the value function to be non-negative.

$$g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_{j,i}} \exp\left[-\frac{1}{2\sigma_{j,i}^2}(x_i - \mu_{j,i})^2\right]. \tag{4.2}$$

The vectors and matrices in the above equations are defined as

$$\begin{aligned} \mathbf{w}(k) &= [w_1(k), w_2(k), \cdots, w_{N_G}(k)]^T \in \mathbb{R}^{N_G \times 1}, \\ \mathbf{h}(\mathbf{x}) &= [g(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), g(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2), \cdots, g(\mathbf{x}, \boldsymbol{\mu}_{N_G}, \boldsymbol{\Sigma}_{N_G})]^T \in \mathbb{R}^{N_G \times 1}, \\ \boldsymbol{\mu}_j &= [\mu_{j,i}] \in \mathbb{R}^{N_G \times n}, \\ \boldsymbol{\Sigma}_j &= [\sigma_{j,i}] \in \mathbb{R}^{N_G \times n}, \end{aligned} \tag{4.3}$$

where k is the iteration index of the policy iteration algorithm introduced next. The covariance matrix $\mathbf{\Sigma}_j$ is taken to be diagonal.

4.2.2 Implementation of RBFNN

In this study, the Gaussian functions are adopted as kernels to approximate an unknown function as is commonly done in statistics and mesh-free finite elements [64–66] That is to say, a domain $\mathcal{D}$ of interest in the state space can be divided into grids and take the grid coordinates as the mean $\boldsymbol{\mu}_j$. Furthermore, the covariance matrix $\mathbf{\Sigma}_j$ is taken to be diagonal and the same constant for all Gaussian functions. It is noteworthy that with these choices of the mean and covariance matrix can indeed deliver sufficiently accurate solutions of nonlinear partial differential equations [67–69]. It is shown in this chapter that the optimal control for nonlinear dynamic systems can also be obtained in this manner, which allows us to focus on the issues in the control design.

Future efforts will treat both $\boldsymbol{\mu}_j$ and $\mathbf{\Sigma}_j$ as trainable coefficients. The effect of varying means and standard deviations on the optimal control solution will be studied in a separate paper.

4.2.3 Policy Iteration (PI) Algorithm

Recall the optimal control and consider the RBFNN solution of the value function in Equation (4.1). At the k^{th} step of iterations, we have

$$\mathbf{u}(\mathbf{x}, k) = -\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})\frac{\partial V(\mathbf{x}, \mathbf{w}(k))}{\partial \mathbf{x}}, \tag{4.4}$$

where $\mathbf{u}(\mathbf{x}, k)$ and $V(\mathbf{x}(t), \mathbf{w}(k))$ denote the control and value function at the k^{th} step of iterations. $\mathbf{u}(t, k = 0)$ is chosen to be a stabilizing initial control. The value function is updated with the following equation,

$$\frac{\partial V^T(\mathbf{x}, \mathbf{w}(k+1))}{\partial \mathbf{x}}[\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}(\mathbf{x}, k)] + \mathcal{L}(\mathbf{x}, \mathbf{u}(\mathbf{x}, k)) = 0. \tag{4.5}$$

According to Theorem 4 of reference [70], the pair $\{\mathbf{u}(\mathbf{x}, k), V(\mathbf{x}, \mathbf{w}(k))\}$ determined by Equations (4.4) and (4.5) leads to a convergent sequence such that

$$\begin{aligned} V(\mathbf{x}, \mathbf{w}(k)) &\geq V(\mathbf{x}, \mathbf{w}(k+1)), \\ \lim_{k\to\infty} V(\mathbf{x}, \mathbf{w}(k)) &= V^*(\mathbf{x}), \\ \lim_{k\to\infty} \mathbf{u}(\mathbf{x}, k) &= \mathbf{u}^*(\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})\frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}}. \end{aligned} \tag{4.6}$$

where $V^*(\mathbf{x})$ denotes the true optimal value function and $\mathbf{u}^*(\mathbf{x})$ is the true optimal control. Given a random value function and iteratively updating the new value function with the greedy policy is called value iteration or successive approximation [70]. Instead of updating the value function, the policy iteration is defined as iteratively updating the greedy policy to the optimal solution. Since the control $\mathbf{u}(x, k)$ and the value function are both iteratively updated during the calculation of the RBFNN, it is hard to distinguish it as value iteration or policy iteration. In this case, it is treated as policy iteration. The proof of convergence of the similar algorithm for saturated controls can be found in [16,71]. Many examples reported in the literature also confirm the convergence of the successive approximation algorithm [17].

Next, we shall present the updating equation in matrix form. Consider the partial derivative,

$$\begin{aligned} \frac{\partial V(\mathbf{x}, \mathbf{w}(k))}{\partial x_i} &= \sum_{j=1}^{N_G} \frac{-(x_i - \mu_{j,i})}{\sigma_{j,i}^2} w_j(k) g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &\equiv \sum_{j=1}^{N_G} d_{i,j}(\mathbf{x}) w_j(k), \end{aligned} \tag{4.7}$$

where

$$d_{i,j}(\mathbf{x}) = \frac{-(x_i - \mu_{j,i})}{\sigma_{j,i}^2} g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j). \tag{4.8}$$

Hence, we have the gradient of the value function as

$$\frac{\partial V(\mathbf{x}, \mathbf{w}(k))}{\partial \mathbf{x}} = \mathbf{D}(\mathbf{x})\mathbf{w}(k) \in \mathbb{R}^{n\times 1}, \tag{4.9}$$

where $\mathbf{D} = [d_{i,j}] \in \mathbb{R}^{n\times N_G}$. The control can now be written in the matrix form as

$$\mathbf{u}(\mathbf{x}, k) = -\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})\mathbf{D}(\mathbf{x})\mathbf{w}(k). \tag{4.10}$$

Define another matrix as

$$\mathbf{G}(\mathbf{x}) = \mathbf{g}\mathbf{R}^{-1}\mathbf{g}^T(\mathbf{x})\mathbf{D}(\mathbf{x}) \in \mathbb{R}^{n\times N_G}. \tag{4.11}$$

Then, the control term in the equation of motion reads

$$\mathbf{g}(\mathbf{x})\mathbf{u}(\mathbf{x}, k) = -\mathbf{G}(\mathbf{x})\mathbf{w}(k). \tag{4.12}$$

Consider the control term in $\mathcal{L}(\mathbf{x}, \mathbf{u})$.

$$\frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}(\mathbf{x}, k) = \frac{1}{2}\mathbf{w}^T(k)\mathbf{H}(\mathbf{x})\mathbf{w}(k), \tag{4.13}$$

where

$$\mathbf{H}(\mathbf{x}) = \mathbf{D}^T \mathbf{g} \mathbf{R}^{-1} \mathbf{g}^T \mathbf{D}(\mathbf{x}) \in \mathbb{R}^{N_G \times N_G}. \tag{4.14}$$

Substituting all the matrix notations to the updating equation (4.5), we arrive at the equation for updating the coefficient vector $\mathbf{w}(k+1)$ as

$$\mathbf{S}^T(\mathbf{x})\mathbf{w}(k+1) + \mathcal{L}(\mathbf{x}, \mathbf{u}(\mathbf{x}, k)) = 0, \tag{4.15}$$

where $\mathbf{S} \in \mathbb{R}^{N_G \times 1}$ and

$$\begin{aligned} \mathbf{S}^T(\mathbf{x}) &= [\mathbf{f}(\mathbf{x}) - \mathbf{G}(\mathbf{x})\mathbf{w}(k)]^T \mathbf{D}(\mathbf{x}) \\ &= \mathbf{f}^T(\mathbf{x})\mathbf{D}(\mathbf{x}) - \mathbf{w}^T(k)\mathbf{G}^T(\mathbf{x})\mathbf{D}(\mathbf{x}) \qquad (4.16) \\ \mathcal{L}(\mathbf{x}, \mathbf{u}(\mathbf{x}, k)) &= \frac{1}{2}\mathbf{x}\mathbf{Q}\mathbf{x}(t) + \frac{1}{2}\mathbf{w}^T(k)\mathbf{H}(\mathbf{x})\mathbf{w}(k). \qquad (4.17) \end{aligned}$$

Define the error of the HJB equation due to the neural networks approximation as

$$e(\mathbf{x}, \mathbf{w}(k+1)) = \mathbf{S}^T(\mathbf{x})\mathbf{w}(k+1) + \mathcal{L}(\mathbf{x}, \mathbf{u}(\mathbf{x}, k)). \tag{4.18}$$

An integrated squared error in the state space $\mathbb{R}^{n \times 1}$ is defined as

$$J_{\mathrm{HJB}}(\mathbf{w}(k+1)) = \int\limits_{\mathbb{R}^{n \times 1}} \frac{1}{2} e^2(\mathbf{x}, \mathbf{w}(k+1)) d\mathbf{x}. \tag{4.19}$$

4.2.4 Sampling Method for Integration

Computation of $J_{\mathrm{HJB}}(\mathbf{w}(k+1))$ involves integration in high dimensional space and can be intensive. Instead of integration, we can uniformly sample a large number of points $\mathbf{x}_s \in \mathcal{D} \subset \mathbb{R}^{n \times 1}$ to compute an approximate value of $J_{\mathrm{HJB}}(\mathbf{w}(k+1))$. We obtain a sampled value J_s of J_{HJB} as

$$\begin{aligned} J_s(\mathbf{w}(k+1)) &= \frac{1}{2}\sum_{s=1}^{N_s} e^2(\mathbf{x}_s, \mathbf{w}(k+1)) \qquad (4.20) \\ &= \frac{1}{2}\sum_{s=1}^{N_s} [\mathbf{S}^T(\mathbf{x}_s)\mathbf{w}(k+1) + \mathcal{L}(\mathbf{x}_s, \mathbf{u}(\mathbf{x}_s, k))]^2 \\ &= \frac{1}{2}\sum_{s=1}^{N_s} \left[\mathbf{w}^T(k+1)\mathbf{S}(\mathbf{x}_s)\mathbf{S}^T(\mathbf{x}_s)\mathbf{w}(k+1) \right. \\ &\left. + 2\mathcal{L}(\mathbf{x}_s, \mathbf{u}(\mathbf{x}_s, k))\mathbf{S}^T(\mathbf{x}_s)\mathbf{w}(k+1) + \mathcal{L}^2(\mathbf{x}_s, \mathbf{u}(\mathbf{x}_s, k))\right], \end{aligned}$$

where N_s is the number of sampled points $\mathbf{x}_s$ in the domain $\mathcal{D}$. Define the following matrix and vectors as

$$\mathbf{A}_g(k) = \mathbf{A}_g^T(k) = \sum_{s=1}^{N_s} \mathbf{S}\mathbf{S}^T(\mathbf{x}_s) \in \mathbb{R}^{N_G \times N_G} \tag{4.21}$$

$$\mathbf{b}(k) = \sum_{s=1}^{N_s} \mathcal{L}(\mathbf{x}_s, \mathbf{u}(\mathbf{x}_s, k))\mathbf{S}(\mathbf{x}_s) \in \mathbb{R}^{N_G \times 1} \tag{4.22}$$

$$d(k) = \frac{1}{2}\sum_{s=1}^{N_s} \mathcal{L}^2(\mathbf{x}_s, \mathbf{u}(\mathbf{x}_s, k)) \in \mathbb{R}^1 \tag{4.23}$$

The sampled performance index reads

$$J_s(\mathbf{w}(k+1)) = \frac{1}{2}\mathbf{w}^T(k+1)\mathbf{A}_g(k)\mathbf{w}(k+1) + \mathbf{w}(k+1)^T\mathbf{b}(k) + d(k). \tag{4.24}$$

The optimal coefficient vector at each iteration step can be obtained as

$$\mathbf{w}(k+1) = -\mathbf{A}_g^{-1}(k)\mathbf{b}(k) \tag{4.25}$$

assuming that the inverse of the matrix $\mathbf{A}^{-1}(k)$ exists. It has been found that when $N_s > N_G$, the inverse usually exists [67–69].

4.2.5 Dominant System

If the policy iteration algorithm starts with an initial estimate of the value function in terms of the RBFNN with randomly generated coefficients, the resulting control can push the system on to a track leading to instability for unstable systems. Hence, it is recommended to begin the policy iterations of an unstable system with a stabilizing initial control, which will ensure system stability through the iteration process. We introduce the concept of dominant system to create the stabilizing initial control. The dominant system is defined as an unstable linear system that has poles with larger positive real parts than that of the original system. A LQR control is employed to stabilize the dominant system and is used as the initial control to start the policy iteration. There are different ways to design the initial control, such as the supervised learning based control [72] and the unconstrained LQR control [16]. It should be noted that for stable systems, randomly generated initial coefficients $\mathbf{w}(1)$ for the RBFNN always make the policy iteration convergent.

Then, the issue is how to estimate the unstable poles of linear systems or unstable dynamics of nonlinear systems. If the mathematical model is available, this information can be extracted from the model. If the model is not available or reliable, we can estimate the Lyapunov exponents of the system from the experimental data.

4.3 Numerical Examples

In the following, we present numerical examples of linear and nonlinear optimal control problems to illustrate the proposed RBFNN solution method. Since it is very difficult to obtain or simulate optimal controls for general nonlinear dynamic systems, we use the linear quadratic regulator suboptimal control as a benchmark to check the accuracy of the RBFNN solution for linear and nonlinear dynamic systems. We first review the suboptimal LQR control.

4.3.1 Linear Systems

Reconsider the linear quadratic regulator (LQR) design defined in the following. The state equation of the system is linear.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}. \tag{4.26}$$

The value function is quadratic given by

$$V(\mathbf{x}(t)) = \frac{1}{2}\int_{t}^{\infty} \left[\mathbf{x}^T(\tau)\mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\mathbf{R}\mathbf{u}(\tau)\right] d\tau = \frac{1}{2}\mathbf{x}^T(t)\mathbf{S}\mathbf{x}(t). \tag{4.27}$$

where $\mathbf{S}$ is the symmetric matrix solution of the algebraic Riccati equation [32]

$$\mathbf{S}\mathbf{A} + \mathbf{A}^T\mathbf{S} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S} + \mathbf{Q} = 0 \tag{4.28}$$

The optimal control is given by

$$\mathbf{u}(t) = -\mathbf{R}^{-1}\mathbf{B}^T \frac{\partial V(\mathbf{x}(t))}{\partial \mathbf{x}} = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}\mathbf{x} \tag{4.29}$$

This is a general solution for any dimension n of the state space. Now, we apply the RBFNN solution method to compute the value function and the optimal control for linear and nonlinear dynamic systems and compare with this optimal control solution.

4.3.2 First-Order System

Let us first consider a first-order system to illustrate the solution process of the RBFNN method.

$$\dot{x} = -ax + f(x) + bu, \quad x(0) = x_0, \tag{4.30}$$

where a and b are constant parameters of the system, and $f(x)$ is a nonlinear function of x. x_0 is an initial condition. Consider a performance index,

$$J = \int_0^\infty \mathcal{L}(x,u)dt \tag{4.31}$$

where $\mathcal{L}(x,u) = \frac{1}{2}(xQx + uRu)$, $Q \geq 0$ and, $R > 0$. Recall the Bellman equation,

$$\frac{\partial V(x)^T}{\partial x}[-ax + f(x) + bu] + \mathcal{L}(x,u) = 0 \tag{4.32}$$

For the linear system $f(x) = 0$, the optimal control is given by Equation (4.29). For the general nonlinear system $f(x) \neq 0$, an explicit analytical expression of the optimal control is hard to find. Here, we apply the RBFNN method to solve the HJB equation to find the optimal control for both linear and nonlinear systems. We have chosen $Q = 10$, $R = 1$ and the number of neurons $N_G = 30$.

The number of sampling points N_s is chosen to be larger than the number of neurons N_G to ensure the invertibility of the matrix A in Equation (4.22). For different systems, the sampling region X_s is larger than the maximum amplitude of the system response while the region for the mean of Gaussian functions X_g should cover the sampling region X_s.

4.3.2.1 Case 1: Linear 1D System

$$\begin{aligned} \dot{x} &= -x + u, \\ x(0) &= x_0, \end{aligned} \tag{4.33}$$

The sampling region for case 1 is chosen as $X_s \in [-5, 5]$ and the region for the mean of Gaussian functions is $X_g \in [-6, 6]$. The weights converge to the optimal solution after 15 iterations and the performance index reaches 10^{-10}. Figure 4.1 shows the results comparison between the RBFNN control and LQR control. From Figure 4.1 we can see that for linear one-dimensional (1D) system, the RBFNN finds exactly same optimal control law as LQR. The bottom figure in Figure 4.1 indicates that the RBFNN finds a global solution in terms of spatial domain x. This can also be validated through investigating the system response with different initial conditions, the results are shown in Figure 4.2. With initial condition ranges from $[-20, 20]$, which is far beyond the sampling region of X_s, the system under RBFNN control exponentially asymptotically converges to the equilibrium point $x^e = 0$.

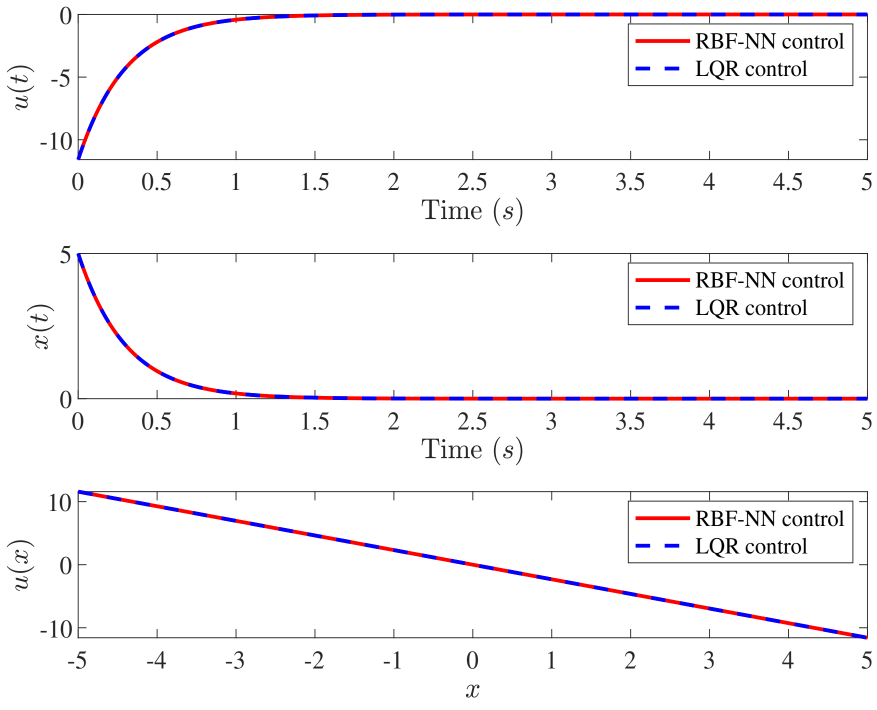


Figure 4.1: Top: Comparison of control law $u(t)$ between RBFNN control and LQR for the linear 1D system. Middle: Comparison of system response $x(t)$ between RBFNN and LQR. The initial condition $x(0) = 5$. Bottom: Comparison of control law u in spatial domain x between RBFNN and LQR.

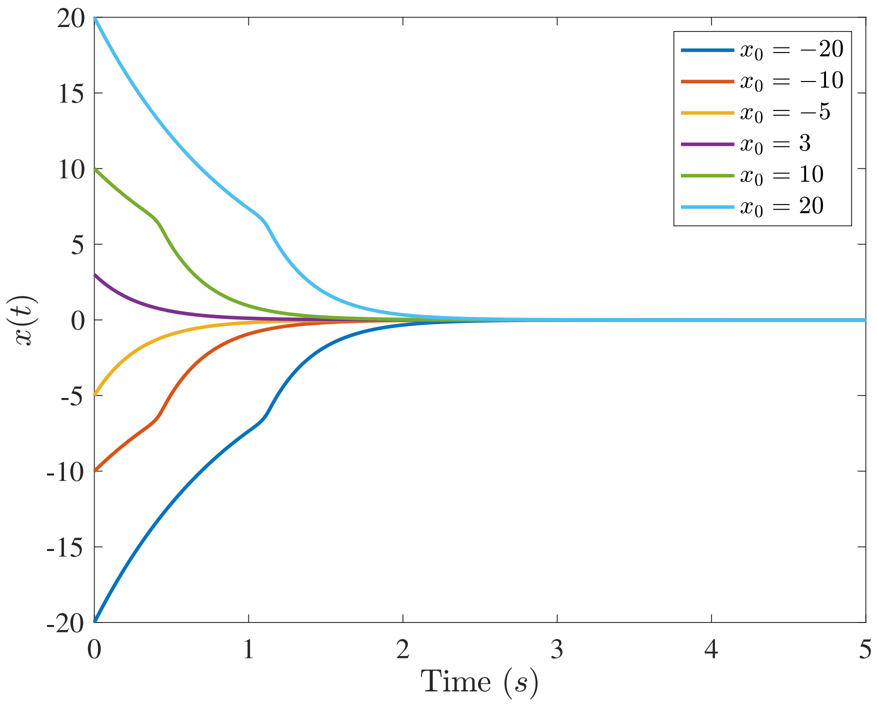


Figure 4.2: Performance of RBFNN control with different initial conditions for the linear 1D system.

4.3.2.2 Case 2: Nonlinear 1D System

$$\begin{aligned} \dot{x} &= x + 10\sin(x) + u, \\ x(0) &= x_0, \end{aligned} \tag{4.34}$$

The sampling region for case 2 is chosen as $X_s \in [-11, 11]$ and the region of neural networks is $X_g \in [-12, 12]$. Q and R are same as the linear 1D system. Figure 4.3 shows the system responses under RBFNN control and LQR control. Here LQR control is designed based on the linear system without nonlinear term. Figure 4.4 shows the system responses under RBFNN control and LQR control. LQR control is designed based on the linearized nonlinear system. We can see that for the first case LQR control cannot stabilize the system under certian initial conditions and for the second case LQR control can stabilize the nonlinear system, however, there is a control saturation during the stabilization. Compared to LQR control in 4.4, RBFNN control doesn't have serious control saturaion and can stabilize the sysetm in a short time. The RBFNN control finds the nonlinear relationship between the control law u and state space x, which gives us the optimal control solution for nonlinear system shown in the bottom in Figure 4.3. For nonlinear system, different initial conditions in the region of sampling points X_s are chosen to validate the performance of the optimal control. From Figure 4.5 we can see that for different initial conditions in X_s, the optimal control performs well to stabilize the system.

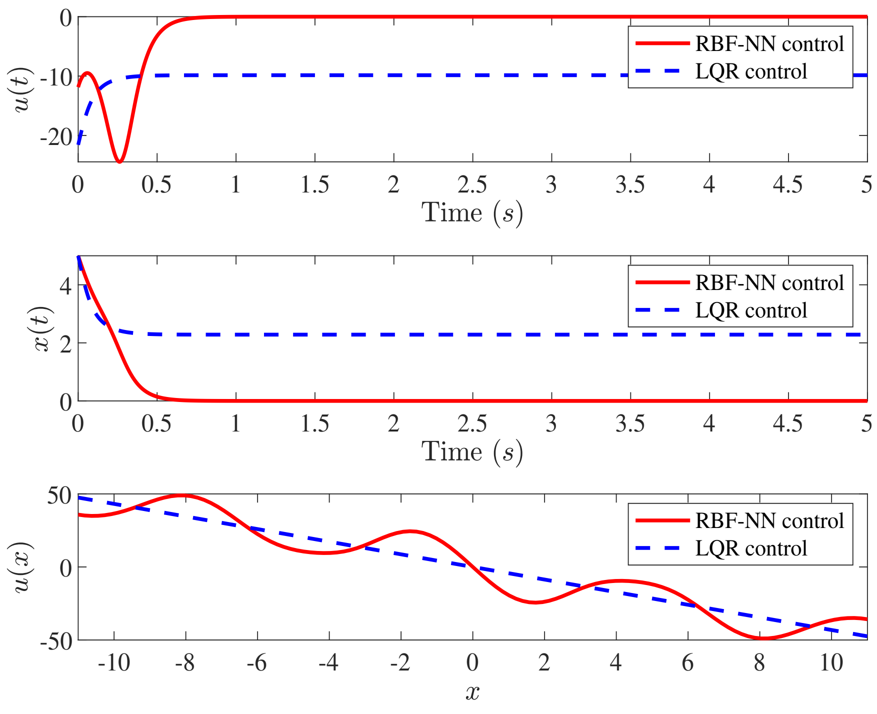


Figure 4.3: Top: Comparison of control law $u(t)$ between RBFNN control and LQR for the nonlinear 1D system. Middle: Comparison of system response $x(t)$ between RBFNN control and LQR. The initial condition $x(0) = 5$. Bottom: Comparison of control law u in spatial domain x between RBFNN control and LQR. Here LQR is designed based on linear system without considering the nonlinear terms.

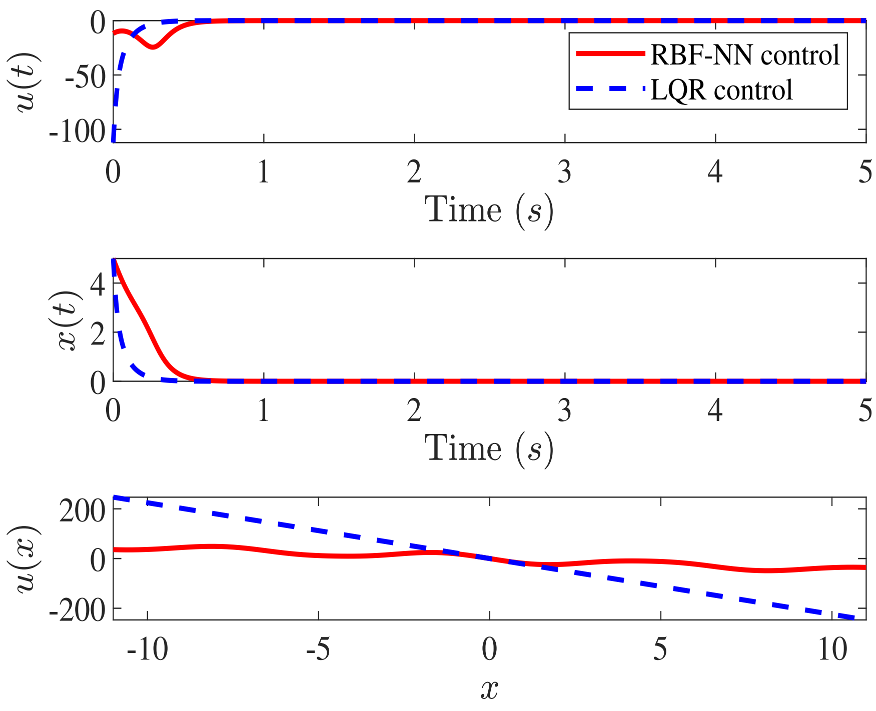


Figure 4.4: Top: Comparison of control law $u(t)$ between RBFNN control and LQR for the nonlinear 1D system. Middle: Comparison of system response $x(t)$ between RBFNN control and LQR. The initial condition $x(0) = 5$. Bottom: Comparison of control law u in spatial domain x between RBFNN control and LQR. Here LQR is designed based on the linearized nonlinear system.

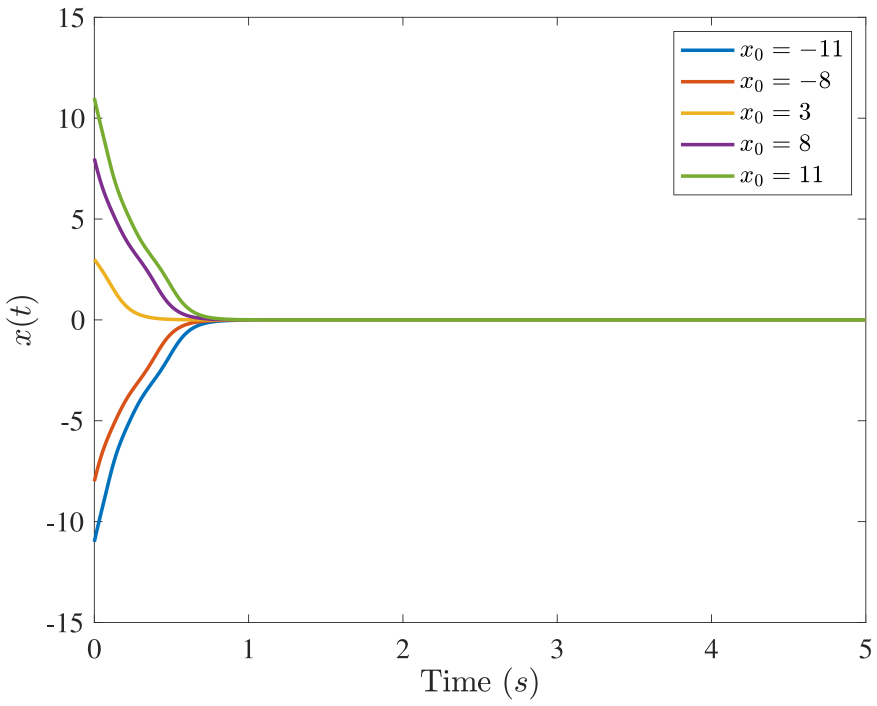


Figure 4.5: Performance of RBFNN control with different initial conditions for the nonlinear 1D system.

4.3.3 Second-Order System

4.3.3.1 Case 3: Linear 2D System

Consider a linear 2D system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{4.35}$$

where $m = 1$, $c = 2$ and $k = 1$. The neural networks region is chosen as $X_g \in [-2, 2] \times [-6, 6]$ and sampling region is chosen as $X_s \in [-1, 1] \times [-3, 3]$. The number of neurons $N_G \times N_G$ are 9×9 and the sampling points are $2N_G \times 2N_G$ along the state space vectors. $\mathbf{Q}$ and $\mathbf{R}$ are chosen as,

$$\mathbf{Q} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \mathbf{R} = 1 \tag{4.36}$$

The results are shown from Figure 4.6 to Figure 4.8. From Figure 4.6 and Figure 4.7 we can see that RBFNN can exactly find the same optimal solution as LQR for 2D system. The value function is shown in Figure 4.8.

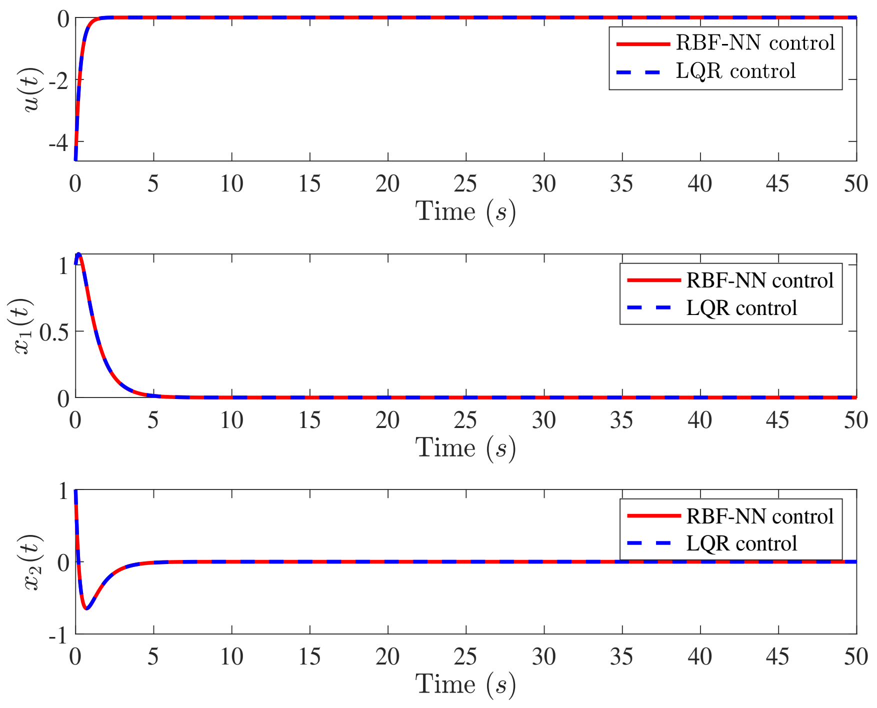


Figure 4.6: Top: Comparison of control law $u(t)$ between RBFNN control and LQR for the linear 2D system. Middle: Comparison of system response $x_1(t)$ between RBFNN control and LQR. Bottom: Comparison of system response $x_2(t)$ between RBFNN control and LQR. The initial condition of the system is $\boldsymbol{x}(0) = [1, 1]^T$.

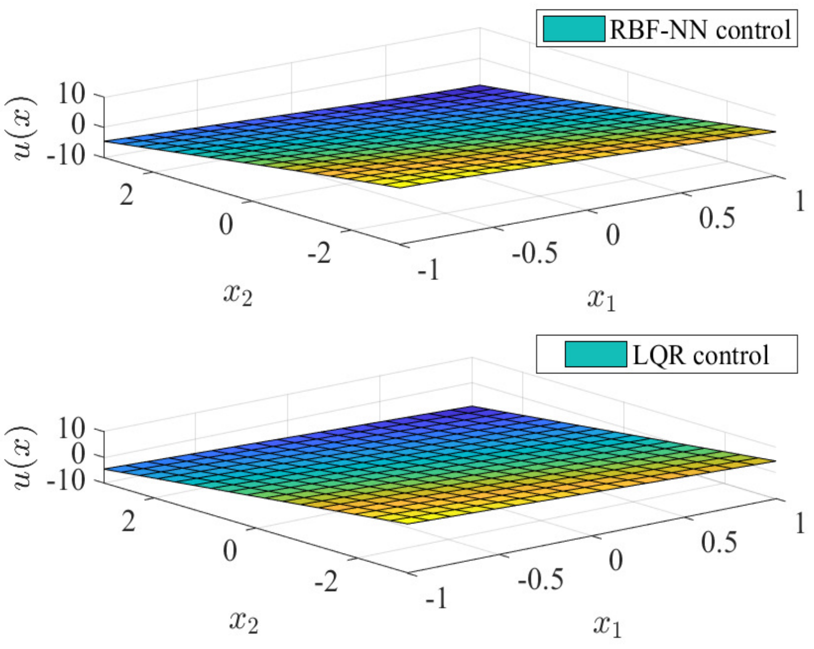


Figure 4.7: Comparison of control law u in spatial domain x between RBFNN control and LQR for the linear 2D system.

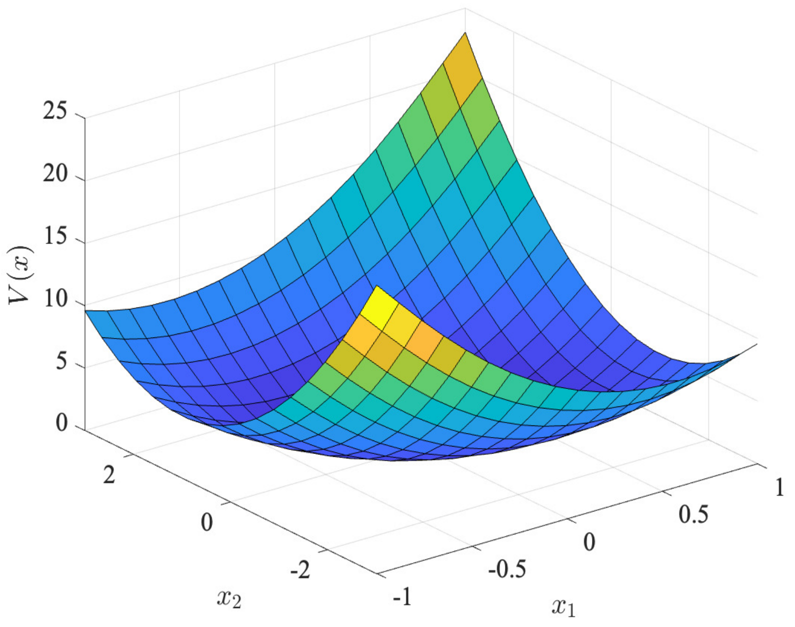


Figure 4.8: Value function $V(x)$ from RBFNN control of the linear 2D system.

4.3.3.2 Case 4: Nonlinear 2D System

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \beta \end{bmatrix} \sin x_1 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{4.37}$$

where $m = 1$, $c = 2$, $k = 1$ and $\beta = 4$. The neural networks region is chosen as $X_g \in [-8, 8] \times [-9, 9]$ and sampling region is chosen as $X_s \in [-5, 5] \times [-6, 6]$. The number of neurons $N_G \times N_G$ are 9×9 and the sampling points are $2N_G \times 2N_G$ along the state space vectors. $\mathbf{Q}$ and $\mathbf{R}$ are chosen same as the linear 2D system.

From Figure 4.9 to Figure 4.10 we can see that RBFNN find the nonlinear optimal solution to stabilize the system. Compared to LQR, RBFNN find the control law $u(x)$ as nonlinear function of the system state $\mathbf{x}$. The value function is shown in Figure 4.11.

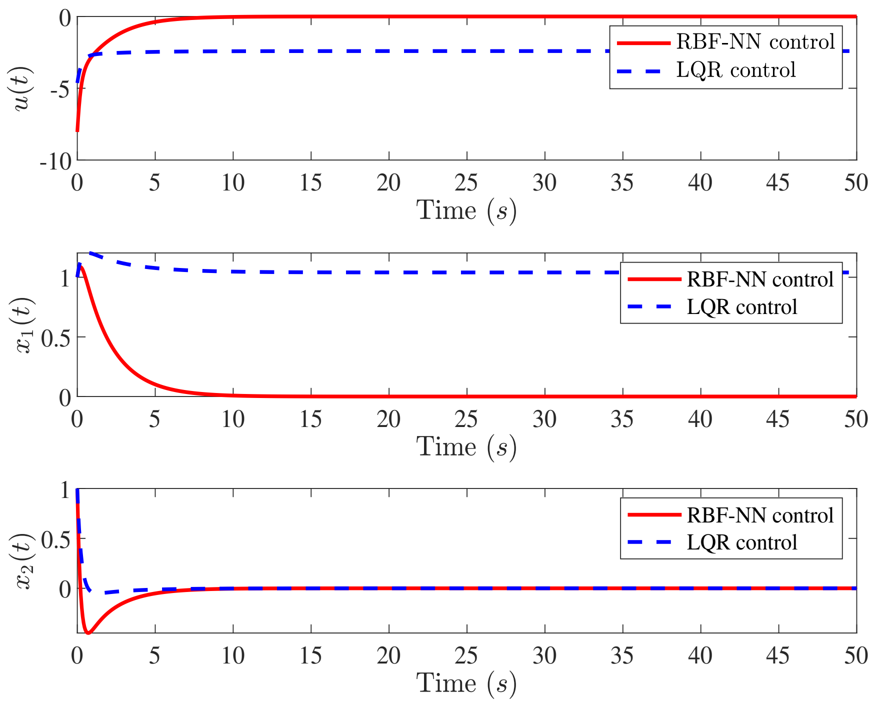


Figure 4.9: Top: Comparison of control law $u(t)$ between RBFNN control and LQR for the nonlinear 2D system. Middle: Comparison of system response $x_1(t)$ between RBFNN control and LQR. Bottom: Comparison of system response $x_2(t)$ between RBFNN control and LQR. The initial condition of the system is $\boldsymbol{x}(0) = [1, 1]^T$.

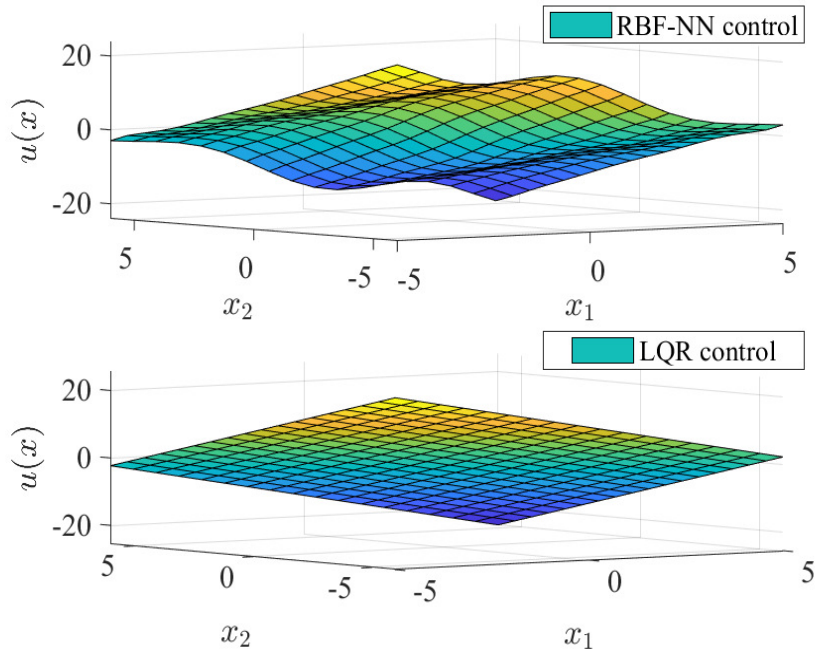


Figure 4.10: Comparison of control law u in spatial domain x between RBFNN control and LQR for the nonlinear 2D system.

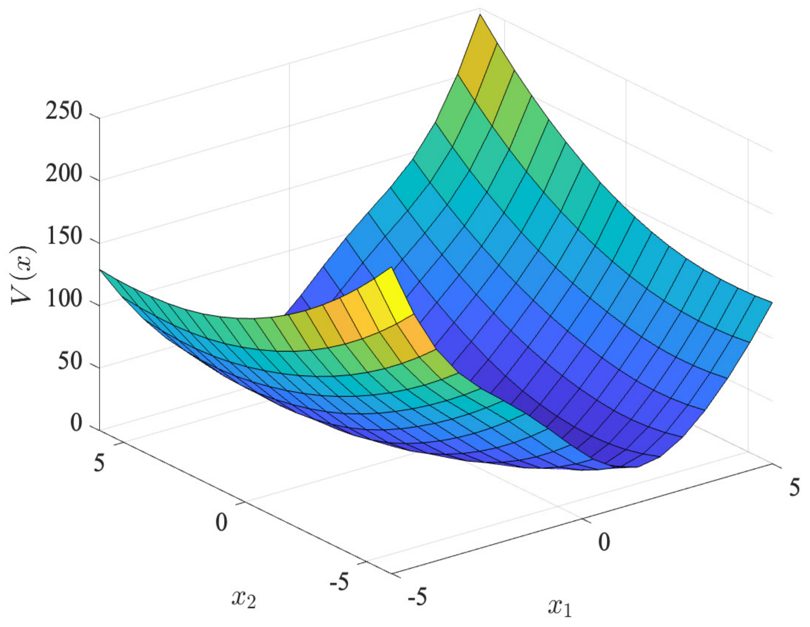


Figure 4.11: Value function $V(x)$ from RBFNN control of the nonlinear 2D system.

All the simulations are done on a ALIENWARE running Windows 10 with 32MB of RAM. With a limited number of neurons, the two-layer RBFNN is powerful enough to find the solution of the optimal control in a very short time. Without using a deep neural networks with plenty of parameters, RBFNN shows high efficiency and good performance to find the optimal solution of nonlinear systems.

Remark 4.3.1. The sampling region and neural networks region are chosen based on the properties of the system dynamics. Here the neural networks region needs to be at least 5σ larger than the region of sampling, to guarantee the existence of the matrix inverse $\mathbf{A}^{-1}(k)$.

4.3.4 Robustness of RBFNN

4.3.4.1 A Second Order System

Neural networks with polynomial activation functions (Poly-NN) have been extensively applied to find the solutions of the optimal control problems in reinforcement learning [16, 21–23, 73–77]. In the following, we compare the performance of controls expressed in terms of the RBFNN with Gaussian neurons and the polynomial neural networks (Poly-NN). Consider a second order nonlinear system,

$$\begin{aligned} \dot{x}_1 &= x_1 + x_2 - x_1(x_1^2 + x_2^2) \\ \dot{x}_2 &= -x_1 + x_2 - x_2(x_1^2 + x_2^2) + u \end{aligned} \tag{4.38}$$

We take the Poly-NN reported in [16] to compare with the proposed RBFNN control. No control constraints are imposed in the comparison. Both the neural networks are trained in the sampling region $X_s \in [-1, 1] \times [-1, 1]$ and applied to two regions $X_{s1} \in [-1, 1] \times [-1, 1]$ and $X_{s2} \in [-2, 2] \times [-2, 2]$ to check their ability to provide good control performance beyond the training region. This is a way to study the generalization of the neural networks.

The Poly-NN has 24 neurons where the RBFNN has $N_G = 5 \times 5 = 25$ neurons. The sampling points are $N_s = 2^2 N_G$ for both neural networks. The standard deviations of Gaussian neurons are $\sigma_1 = \sigma_2 = 0.75$. To investigate robustness of the controls, we consider a disturbance $d(t)$ to each state of the system as shown in Figure 4.12.

$$d(t) = \begin{cases} 1 + g(t), & 10 \le t \le 11 \\ 5 + g(t), & 20 \le t \le 21 \\ 10 + g(t), & 30 \le t \le 31 \\ g(t), & \text{else} \end{cases} \tag{4.39}$$

where $g(t)$ is the Gaussian white noise. Its signal-to-noise (SNR) ratio is 50dB.

The closed-loop responses are shown from Figure 4.13. Within the training region $X_s \in [-1, 1] \times [-1, 1]$, the RBFNN, Poly-NN and LQR controls have about

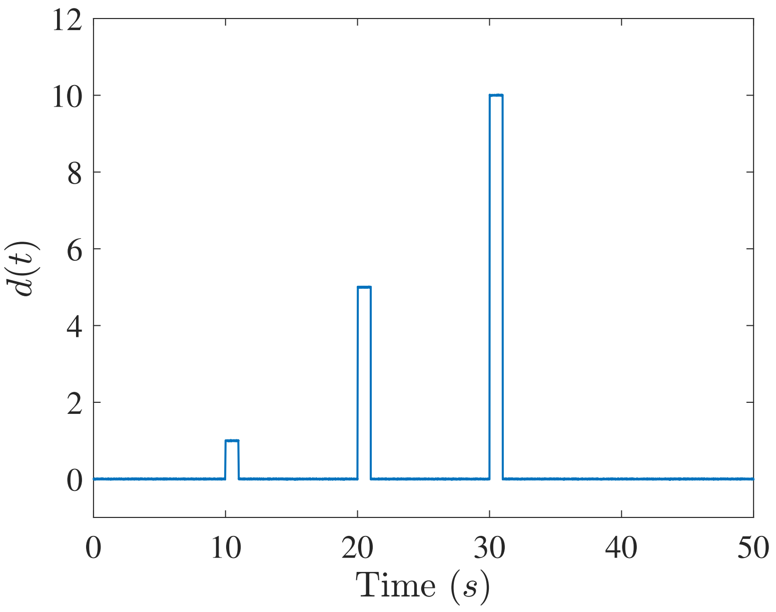


Figure 4.12: Disturbances to the second order nonlinear system.

the same performance when the system starts from the same initial condition $\mathbf{x}_0 = [0.5, 0.5]^T$ and is subject to the disturbance $d(t)$. Note that the LQR control is designed based on the linearized system. When the disturbance becomes stronger, the RBFNN control can still stabilize the system with reasonable effort. However, the Poly-NN control performs poorly.

This is due to the well-known fact that polynomials generate large extrapolation errors in regression applications. In the case of optimal control design, both the RBFNN and Poly-NN are trained with the sampled data in the region $X_{s1} \in [-1, 1] \times [-1, 1]$. When the response of the system moves out of this region, both the RBFNN and Poly-NN controls are based on the extrapolated solution of the HJB equation. It is found that the Poly-NN control does not perform as good as the RBFNN control. It is interesting to show both the controls in the training region $X_{s1} \in [-1, 1] \times [-1, 1]$ and in a larger region $X_{s2} \in [-2, 2] \times [-2, 2]$. As seen from the figure, the Poly-NN control grows unbounded quickly outside the training region and can no longer stabilize the system, while the RBFNN control remains bounded and can still stabilize the system in this larger region. As a matter of fact, the RBFNN control will remain bounded and approach zero at locations far away from the training region, because all the Gaussian neurons with means in the training region decrease exponentially.

Hence, the Poly-NN control is not readily 'generalizable' beyond the training region and thus not robust to large disturbances. The RBFNN with Gaussian neurons obtains nonlinear optimal controls that deliver better performance and robustness to larger disturbances than the Poly-NN can handle, and has better ability to 'generalize' beyond the training region than the Poly-NN.

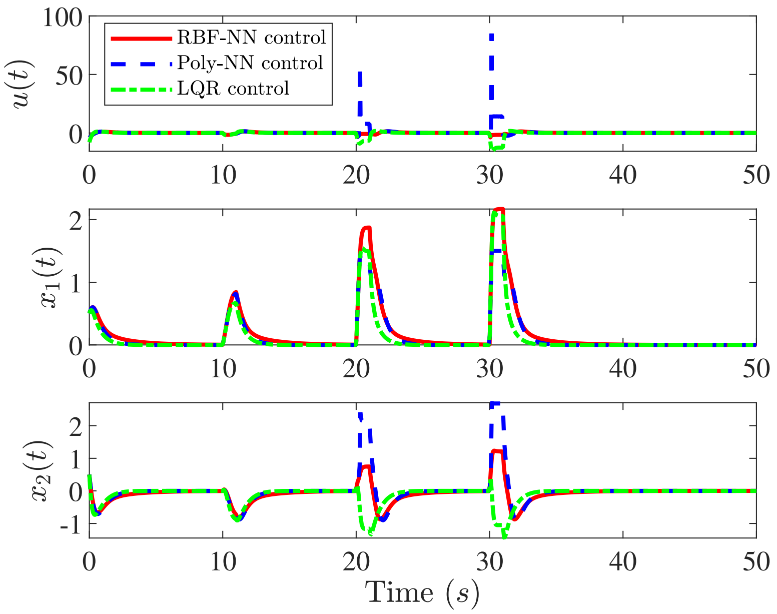


Figure 4.13: Performance comparison of RBFNN, Poly-NN, and LQR controls for the nonlinear system in Equation (4.38). Top: The control $u(t)$. Middle: The response $x_1(t)$. Bottom: The response $x_2(t)$.

4.3.4.2 Duffing System

Consider the Duffing system as another example to test the robustness of RBFNN controller with changing parameters,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ \beta \end{bmatrix} x_1^3 \tag{4.40}$$

The sampling region is $X_s \in [-2, 2] \times [-8, 8]$. Corresponding to the sampling region, the means of Gaussian neurons are uniformly distributed in the region $X_g \in [-3, 3] \times [-9, 9]$.

We take $\beta = 2$ as the nominal value and allow β to vary in a known range $[1, 6]$. Both the RBFNN and LQR controls are designed with the nominal value of

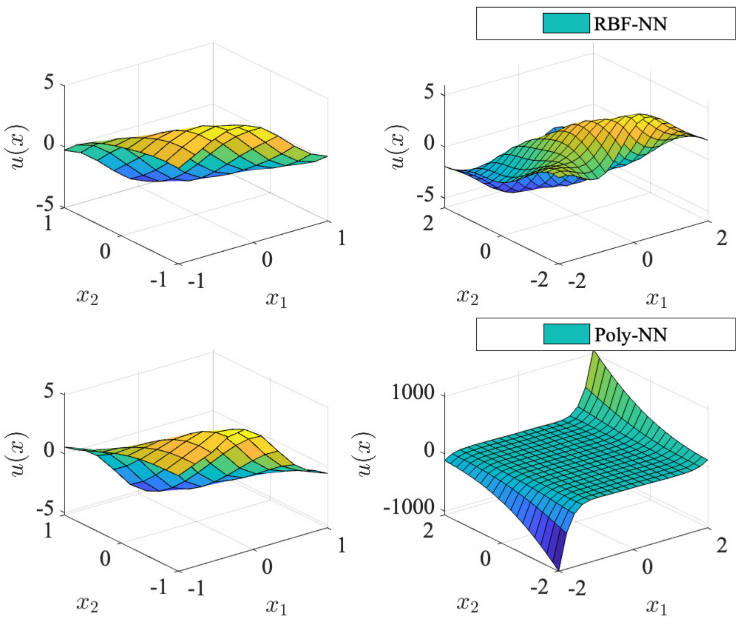


Figure 4.14: Comparison of spatial distribution of RBFNN and Poly-NN optimal controls u as a function of the state $\mathbf{x}$. Left: The control $u(\mathbf{x})$ plotted in the training region $X_{s1} \in [-1,1] \times [-1,1]$. Right: The control $u(\mathbf{x})$ plotted beyond the training region into the larger region $X_{s2} \in [-2,2] \times [-2,2]$.

β. Note that the LQR control is designed for the linearized system at $\beta = 2$. The performance index J of the LQR design is defined as,

$$J(\mathbf{x}, \mathbf{u}) = \frac{1}{2} \sum_{k=0}^{n_s} \left[\mathbf{x}^T(k) \mathbf{Q} \mathbf{x}(k) + R u^2(k) \right]. \tag{4.41}$$

where k is the discrete time index, and n_s is number of integration steps of the simulation. The matrices are given by

$$\mathbf{Q} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 1 \tag{4.42}$$

We have chosen $n_s = 1000$. The integration time step is $\Delta t = 0.01$ seconds. We simulate the closed-loop system starting from the same initial condition $\mathbf{x}_0 = [0.7, 0.7]^T$.

The index J is used to evaluate the performance of both the RBFNN and LQR controls. Figure 4.15 shows the performance index J as a function of β for both the

RBFNN and LQR controls. When the actual value of β is less than 2, the nominal value used in control design, the linearized model of the system remains valid. As a result, the LQR control is slightly better than the RBFNN control. However, when $\beta > 2$, the linearized model at $\beta = 2$ becomes less accurate, resulting in poor performance of the LQR control compared with the RBFNN control. When β reaches a critical value β_{cr}, marked by the vertical lines in the figure, the closed-loop system becomes unstable and the performance index J grows unbounded. For the LQR control, $\beta_{cr} = 4.1579$, while for the RBFNN control, $\beta_{cr} = 4.5263$. This is a quantitative measurement of robustness to model uncertainty.

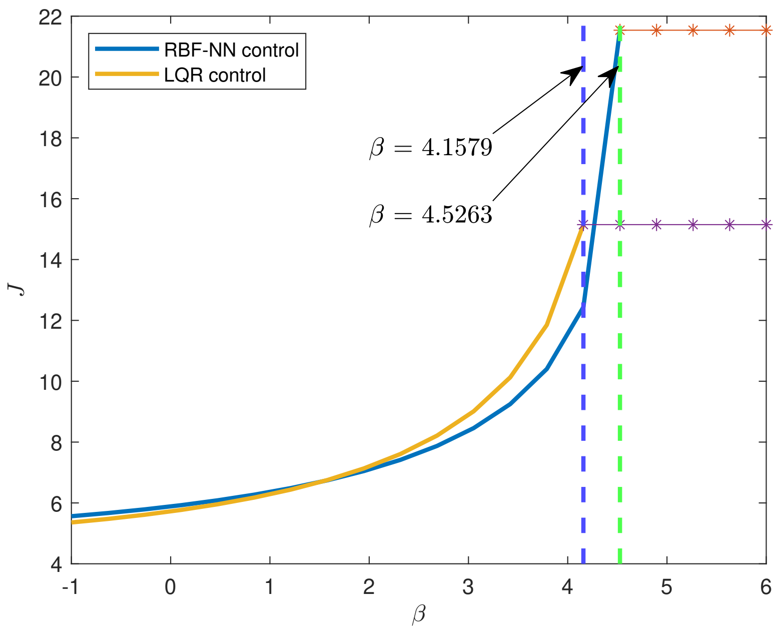


Figure 4.15: Robustness of the RBFNN and LQR controls with respect to the model uncertainty β. The vertical dash lines mark the critical value of β, beyond which the closed-loop system becomes unstable.

Finally, we should point out that the RBFNN with one hidden layer can find optimal controls with good performance and robustness for nonlinear dynamic systems. The design of RBFNN itself is not optimized in this study. However, there are studies on how to design RBFNN in the literature, such as ErrCor algorithm [61], hierarchical growing strategy (GGAP algorithm) [78], MRAN algorithm [79], resource allocation algorithm [80], and adaptive RBF algorithms [81–83]. These algorithms can be considered for optimal design of RBFNN architecture in the future study.

Remark 4.3.2. Numerical examples shown above illustrate that RBFNN show significant performance in terms of finding nonlinear optimal solutions. However, there is still a problem regarding to the design of RBFNN even with one hidden layer. There are some developed algorithms found in literature for RBFNN construction, such as ErrCor algorithm [61], hierarchical growing strategy (GGAP algorithm) [78], MRAN algorithm [79] and so on [80]. Resources about ErrCor algorithm can be found in: `https://www.eng.auburn.edu/~wilambm/nnt/index.htm`. Some adaptive RBF algorithms can also be found in [81–83]. These algorithms can be considered for RBFNN design in the future study.

4.4 RBFNN with Model Reduction and Transfer Learning

In this section, the RBFNN optimal control is combined with model reduction technique to obtain the reduced order model based optimal control. Transfer learning is introduced to compensate the model uncertianty that between the simulated model and the hardware.

4.4.1 RBFNN Optimal Control with Reduced Order Model

Apply the RBFNN optimal control algorithm in Section 4.2.1 to the reduced order system in Equation (3.15), the trainable weights of RBFNN can be obtained and denoted as $\mathbf{w}_r$. Based on Equation (4.10) the RBFNN optimal control for the reduced order model can be expressed as,

$$\mathbf{u}(t) = -\mathbf{R}_r^{-1}\mathbf{B}_r^T\mathbf{D}_g(\mathbf{x}_r(t))\mathbf{w}_r \tag{4.43}$$

where $\mathbf{R}_r$ is the LQR gain matrix for the reduced order model. In the following sections, we illustrate how to implement the trained RBFNN and control law designed with the reduced order system to the original system.

In this section, we discuss how to implement the optimal policy obtained with the reduced order model to the original system. Consider two cases - stabilization and trajectory tracking, which are two typical control problems for mechanical systems.

For stabilization, the control law for the reduced order model can be designed as,

$$\mathbf{u}(t) = -\mathbf{R}_r^{-1}\mathbf{B}_r^T\mathbf{D}_\mathbf{g}(\mathbf{x}_r(t))\mathbf{w}_r \tag{4.44}$$

For trajectory tracking case, at first define a desired trajectory as $\mathbf{x}_{ref_r}(t)$ for the reduced order system. The optimal control problem amounts to finding control $\mathbf{u}$ in an admissible set that drives the system from the initial state $\mathbf{x}_{0r}$ to track the desired trajectory $\mathbf{x}_{ref_r}(t)$. The controller is designed based on the LQR feedback control for linear system and given as,

$$\begin{align} \mathbf{u}(t) &= -\mathbf{K}(\mathbf{x}_r(t) - \mathbf{x}_{ref_r}(t)) \tag{4.45} \\ &= \mathbf{K}\mathbf{x}_{ref_r}(t) - \mathbf{K}\mathbf{x}_r(t) \tag{4.46} \end{align}$$

Recall Equation (4.10). The feedback law of the RBFNN optimal control is given as,

$$\mathbf{u} = -\mathbf{K}\mathbf{x} = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{D}_\mathbf{g}(\mathbf{x})\mathbf{w} \tag{4.47}$$

Substitute Equation (4.47) into Equation (4.46), the control law for the reduced order model can be obtained as,

$$\mathbf{u}(t) = -\mathbf{R}_r^{-1}\mathbf{B}_r^T(\mathbf{D_g}(\mathbf{x}_r(t)) - \mathbf{D_g}(\mathbf{x}_{ref_r}(t)))\mathbf{w}_r \tag{4.48}$$

A block diagram is shown in Figure 4.16 to illustrate the reduced order model based control design.

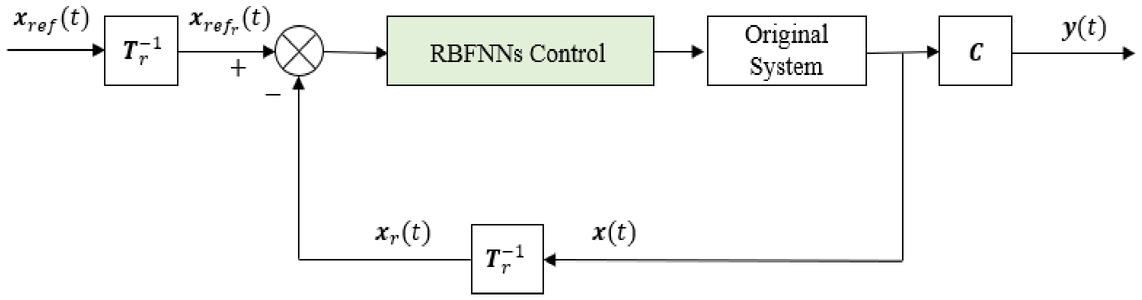


Figure 4.16: Block diagram of the RBFNN optimal control algorithm with model reduction.

4.4.2 RBFNN Optimal Control with Transfer Learning

When the RBFNN control presented in the previous section is well trained using the data simulated with a sufficiently accurate model of the dynamic system, it gives good control performance. When there are model uncertainties, unknown hardware gains and external disturbances, the performance of the model-based RBFNN control may deteriorate. To improve the robustness of the RBFNN control, we make use of the transfer learning concept to retrain the neural networks with the experimental data.

Transfer learning is a machine learning technique for fine tuning network weights [84] for performance improvement when new data become available. It is common to freeze the weights in the hidden layers of the neural networks, and retrain the weights in the output layer with the new data. In this study, we only have one hidden layer whose weights are retrained with the experimental data. This is done by applying the new data to Equation (4.20) and update the network coefficients according to Equation (4.25) during the policy iteration. The system response $\mathbf{x}(t)$ is used to update the system dynamics and performance index. The system response $\mathbf{x}(t)$ is collected from experiments and used to retrain the RBFNN. Note that part of the system dynamics parameters remain model-based during the RBFNN training. The idea is making use of the model of the system and introducing the experimental data to update the performance of the optimal controls. The concept of transfer learning is compatible with this approach, allowing knowledge to be transferred from one domain (model-based) to another (experimental).

The retrained RBFNN control is experimentally evaluated and compared with the original model-based RBFNN control. A flowchart in Figure 4.17 summarizes the RBFNN control design procedure discussed in the previous sections.

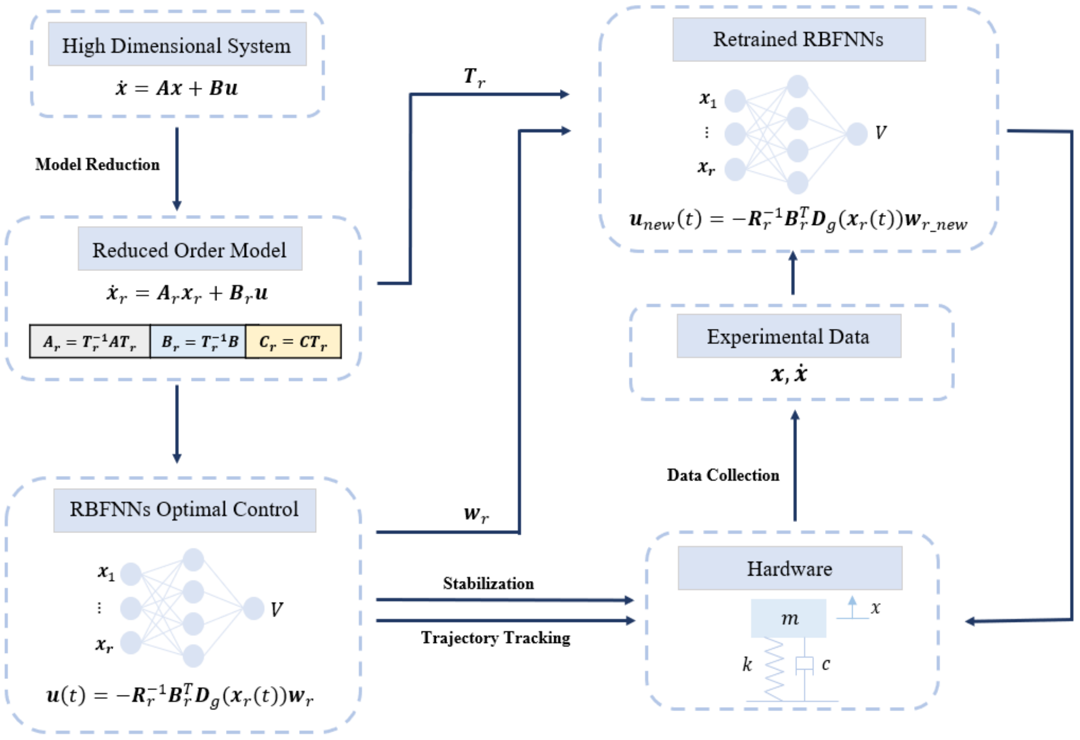


Figure 4.17: Flowchart of the RBFNN optimal control algorithm with model reduction and transfer learning.

4.5 Experimental Validations

4.5.1 Quanser Qube2

In this section, the inverted pendulum is adopted to validate the RBFNN control with a reduced-order model in Section 3.1. The rotary pendulum is a popular dynamic system to test the performance of neural networks controls. Ref. [85] proposes a stable adaptive neural network control scheme to suppress oscillations and compensate external disturbances for rotary inverted pendulum. Meanwhile the rotary pendulum has two degrees of freedom. The state space is four dimensional, which makes the computation of RBFNN control more complicated.

4.5.2 Qube2 with Balanced Truncation

4.5.2.1 Numerical Simulations

The linearized model of the rotary pendulum at the equilibrium point is defined as,

$$J_r\ddot{\theta} + m_p lr\ddot{\alpha} = \tau - b_r\dot{\theta} \tag{4.49}$$

$$J_p\ddot{\alpha} + m_p lr\ddot{\theta} + m_p gl\alpha = -b_p\dot{\alpha} \tag{4.50}$$

where θ is the angle of the rotary arm, α is the angle of the pendulum, τ is the applied torque at the base of the rotary arm. $J_r = m_r r^2/3$ is the moment of inertia of the rotary arm with respect to the axis of rotation of rotary arm, $J_p = m_p L_p^2/3$ is the moment of inertial of the pendulum link relative to the axis of rotation of pendulum, b_r and b_p are the viscous damping of the rotary arm and the pendulum, respectively. m_r is the mass of the rotary arm and m_p is the mass of the pendulum, L_p is the length of the pendulum and $l = L_p/2$. The *nominal values* of the rotary pendulum parameters are listed in Table 4.1. A schematic of the rotary inverted pendulum is shown in Figure 4.18.

Note that the parameters in Table 4.1 are the nominal values provided by Quanser. These nominal parameters and the linearized model in Equation (4.49) and Equation (4.50) may not describe the physical system accurately.

With the linear model in Equation (4.49) and Equation (4.50) and the nominal values of the parameters in Table 4.1, the pendulum model in the state space form is given as,

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 152.0057 & -12.2542 & -0.5005 \\ 0 & 264.3080 & -12.1117 & -0.8702 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 50.6372 \\ 50.0484 \end{bmatrix} \tau \tag{4.51}$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}(t) \tag{4.52}$$

Table 4.1: Parameters of the rotary pendulum system.

Symbol	Description	Value
m_r	Mass of rotary arm	0.095 kg
r	Total length of rotary arm	0.085 m
J_r	Moment inertial of rotary arm	$2.29 \times 10^{-4}\ kg \cdot m^2$
b_r	Viscous damping of rotary arm	$10^{-3} N \cdot m \cdot s/rad$
m_p	Mass of pendulum	0.024 kg
L_p	Total length of pendulum	0.129 m
l	Half length of pendulum	0.0645 m
J_p	Primary lateral stiffness per axle	$1.33 \times 10^{-4}\ kg \cdot m^2$
b_p	Viscous damping coefficient	$5 \times 10^{-5} N \cdot m \cdot s/rad$
g	Secondary lateral stiffness	$9.81\ m \cdot s^2$

where $\mathbf{x} = [\theta, \alpha, \dot{\theta}, \dot{\alpha}]^T$ is the state vector of the rotary pendulum. The outputs are the angles of rotary arm θ and pendulum α. For simulation purpose, it is assumed that all the system states are the outputs to test the performance of the full state feedback controller. In real experiment, the derivatives of the angle measurements can be obtained through a low pass filter given as,

$$H(s) = \frac{50s}{s + 50} \tag{4.53}$$

We apply the balanced truncation model reduction technique in Section 3.1 to obtain the reduced order model of the rotary pendulum. The Hankel singular values of the rotary pendulum are shown in Figure 4.19. The first two columns represent the unstable modes of the system with infinite values, which are actually out of the bounds of the plot. The last two columns represent the stable modes of the system with finite singular values. Figure 4.19 suggests that the first three eigen-modes dominate the system. Hence, the order of the reduced order model is chosen as $r = 3$ to approximate the original model based on the theory of balanced truncation.

The traditional LQR algorithm is applied to the original system as a benchmark to compare with the results from RBFNN. Once the reduced order model is obtained, the LQR gain matrices for the original model and reduced order model

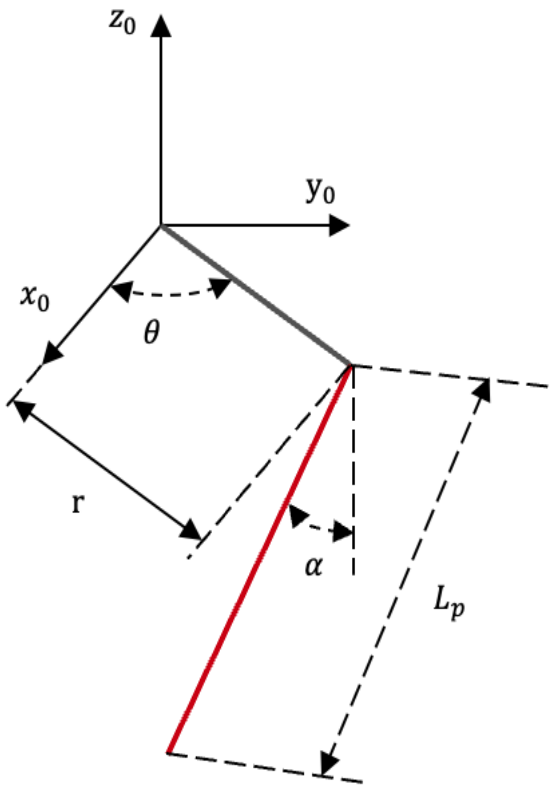


Figure 4.18: Schematic of the rotary inverted pendulum.

are given as,

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{Q}_r = \begin{bmatrix} Q_{11} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R} = \mathbf{R}_r = 1. \tag{4.54}$$

Different values of Q_{11} can be chosen to test the performance of RBFNN. The RBFNN optimal control is implemented once the reduced order model is obtained. The sampling region for the reduced order model is chosen as $X_{1s} \times X_{2s} \times X_{3s} = [-1,1] \times [-1,1] \times [-1,1]$. The region for Gaussian neurons is chosen as $X_{1g} \times X_{2g} \times X_{3g} = [-4,4] \times [-4,4] \times [-4,4]$. The number of neurons $N_G \times N_G \times N_G$ are $4 \times 4 \times 4$ and the sampling points are $2N_G \times 2N_G \times 2N_G$. The trained weights are denoted as w_r for the reduced order model.

Next, we apply the control to the trajectory tracking problem. The reference

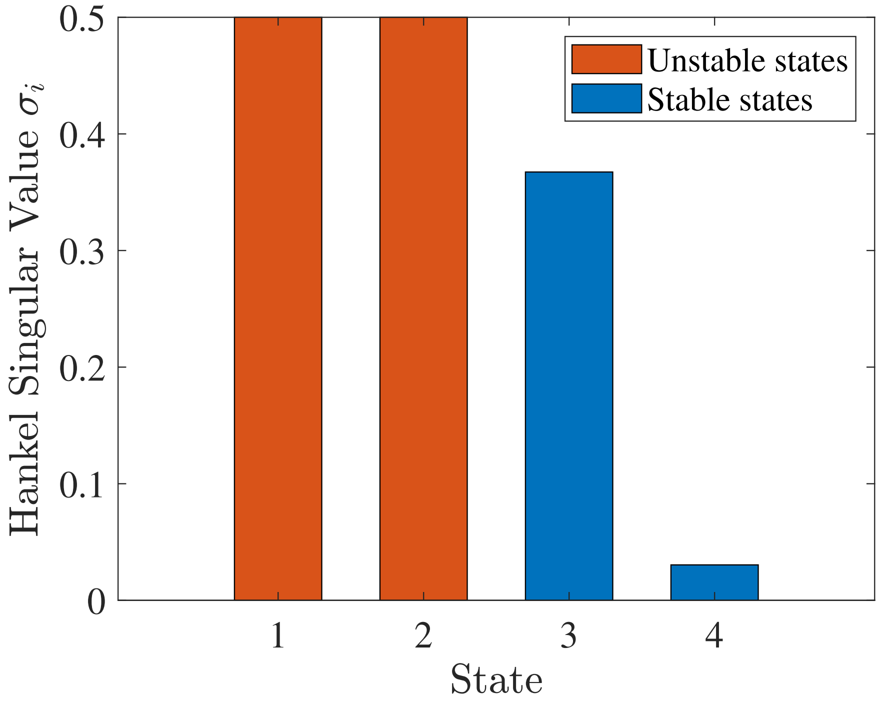


Figure 4.19: Hankel singular values of the rotary pendulum. The first two columns represent the unstable modes of the system with infinite values.

signal is chosen as,

$$\mathbf{x}_{ref} = [\theta_r, 0, 0, 0]^T \tag{4.55}$$

It can be projected onto the reduced order model space through the transformation matrix $\mathbf{T}_r$ in Section 3.1.2,

$$\mathbf{x}_{ref_r} = \mathbf{T}_r^{-1}\mathbf{x}_{ref} \tag{4.56}$$

The optimal control is then given as,

$$\mathbf{u}(t) = -\mathbf{R}_r^{-1}\mathbf{B}_r^T(\mathbf{D}_\mathbf{g}(\mathbf{x}_r(t)) - \mathbf{D}_\mathbf{g}(\mathbf{x}_{ref_r}(t)))\mathbf{w}_r \tag{4.57}$$

A square wave is chosen as θ_r. Figures 4.20 and 4.21 show the tracking performance of the rotary pendulum for the cases when $Q_{11} = 5$ and $Q_{11} = 100$. A root mean square (RMS) error is defined to quantify the performance of the trajectory tracking,

$$e_{RMS} = \sqrt{\frac{1}{n_s}\sum_{i=1}^{n_s}(\theta(i) - \theta_r(i))^2} \tag{4.58}$$

where $n_s = 50000$ in the simulated rotary pendulum system with $t \in [0, 50], \Delta t = 0.001$. From Figure 4.20, we can see that the RBFNN optimal control has the ability to track the desired trajectory and shows similar results compared to LQR. The root mean square errors of the original LQR and RBFNN are shown in Figure 4.22. When $\mathbf{Q}_r$ has the same diagonal value as matrix $\mathbf{Q}$, the RMS of RBFNN is 4.12, which is a little larger than 3.71 for LQR. In this case, the RBFNN controller has less saturation compared to the LQR as shown in Figure 4.20. However, it can still track the desired trajectory with low RMS error. In Figure 4.21, when Q_{11} is increased to 100, the RBFNN controller shows better tracking performance compared to LQR. The RMS error of RBFNN controller is 3.13 which is less than LQR. Meantime RBFNN control shows fast tracking response and comparable control saturation in contrast to LQR.

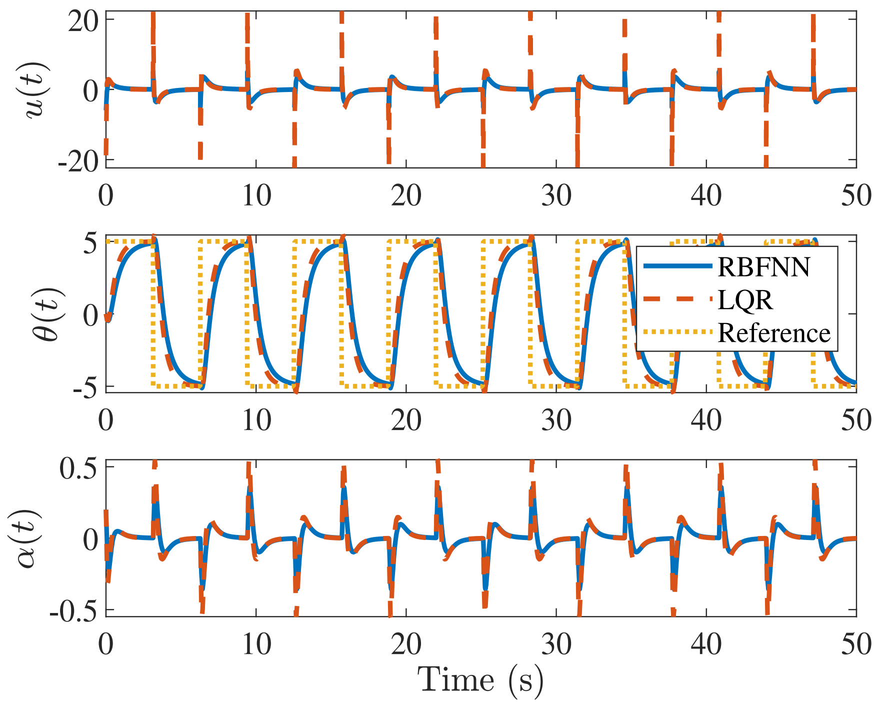


Figure 4.20: Tracking tracking of the rotary pendulum with the reduced order model based RBFNN. $Q_{11} = 5$.

4.5.2.2 A Summary of Simulation Results

In this section we numerically demonstrate that the proposed RBFNN optimal control with reduced order model shows good performance in terms of the trajectory tracking of the rotary pendulum. In the simulation, the RBFNN is pretrained and can be adopted as the initial RBFNN control for the hardware. In the following section, we apply the pre-trained RBFNN control to the Quanser-Servo2

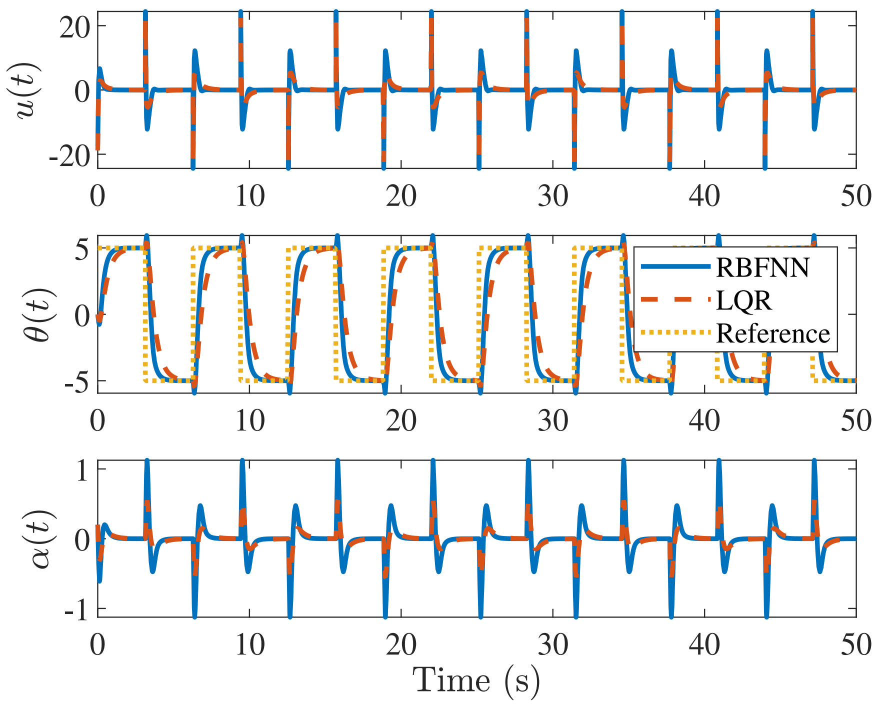


Figure 4.21: Tracking tracking of the rotary pendulum with the reduced order model based RBFNN. $Q_{11} = 100$.

inverted pendulum system to validate its performance and further apply transfer leaning to update the RBFNN.

The rotary pendulum model not only suffers from the curse of dimensionally, but also it is extremely hard to find a good initial stabilizing random control for the rotary pendulum to start the policy iteration for the RBFNN optimal control. This could be due to the unstable equilibrium of the inverted pendulum. It is found that numerically it is much easier to find an initial stabilizer for the reduced order system to make the RBFNN solution converge to the optimal one. This is an interesting phenomenon that deserves further investigations.

4.5.2.3 Experimental Validation

In this section, Quanser-Servo2 inverted pendulum system is adopted to validate the RBFNN optimal control algorithm. The RBFNN optimal control with $Q_{11} = 300$ is first applied to the Quanser-Servo2. To deomonstrate the performance of the RBFNN control, two different cases - pendulum balancing and rotary arm trajectory tracking - are tested. The hardware setup of the Quanser-Servo2 is shown in Figure 4.23. First, we apply RBFNN optimal control for the trajectory tracking. A square wave is chosen as the reference signal for the rotary arm, and the tracking

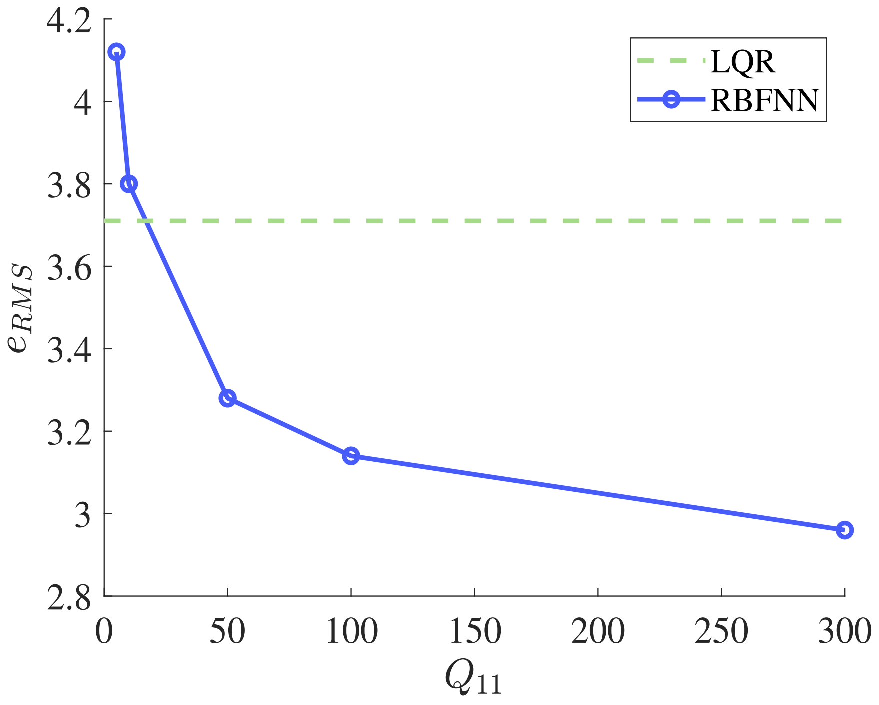


Figure 4.22: Variation of the tracking error of the rotary pendulum with different Q_{11} for the reduced order model based RBFNN. The dash line represents the tracking error of the traditional LQR and serves as a benchmark to illustrate the performance of the reduced order model based RBFNN.

responses from the RBFNN control are shown in Figure 4.24. The RBFNN optimal control shows good performance to track the desired trajectory. With the trajectory tracking experimental data available, the data in the rectangle zone on the figure is selected to retrain the RBFNN offline to improve tracking performance. All the experiments are done with the help of Quanser HIL and MATLAB Simulink with $t \in [0, 60]$, $\Delta t = 0.002$.

We should point it out that the data needs to be carefully selected to retrain the RBFNN. A chosen criterion is the error of the HJB equation during the training. The data from experiments does not satisfy the model-based RBFNN accurately, however, the error of the HJB equation will converge to a certain value as the weights converge.

Quanser adopted Deep Deterministic Policy Gradient (DDPG) [86] in reinforcement learning to balance the inverted pendulum with designed reward function and constrains on the rotary arm, inverted pendulum angle and motor voltage. The nonlinear model of Quanser-Servo2 is used to train the reinforcement learning in simulation considering the safety issue of hardware. To better illustrate the performance

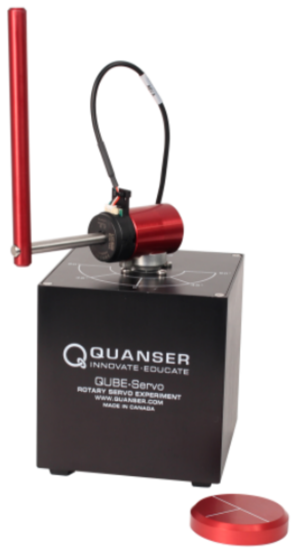


Figure 4.23: Quanser-Servo2 inverted pendulum system hardware [1].

and advantages of the RBFNN optimal control, the Quanser default LQR example, the pre-trained RBFNN in Section 4.5.2.1 and the RL policies from Qunaser [87] are implemented to balance the pendulum and compared with the results from the retrained RBFNN. The controls are designed to stabilize the rotary arm at 0°. The LQR gain matrices are same as in Section 4.5.2.1. There are two different RL policies provided from Quanser, one is denoted as RL and another is denoted as DDPG to follow the file names $QubeIPBalRLPolicy.mat$ and $QubeIPBalDDPG09.mat$ in Quanser QUBE-Servo2 pendulum control reinforcement learning module [88]. The performances are shown in Figures 4.25 and 4.26 in the balancing performance of Quanser-Servo2 inverted pendulum.

Figures 4.25 to 4.26 show that there are two main issues with the RL policies from Quanser. The RL policies are able to balance the pendulum rapidly. However, the balanced position of the rotary arm is at around 25°. It appears that the reference angle of the rotary arm is not included in the reinforcement learning design. Moreover, one of the RL policies shown as dot line in Figure 4.25 is not stable in long time execution. It has large oscillations when $t \in [40, 55]$. The LQR and the pre-trained RBFNN optimal control both are able to balance the pendulum with small oscillations. However, the retrained RBFNN control shows significant suppression of oscillations when balancing the pendulum and stabilize the rotary arm at 0°. There are almost no oscillations for both rotary arm and pendulum in the operation over a long time. The voltages from different algorithms are shown in Figure 4.26. It is obvious to see that the retrained RBFNN control requires less energy to balance the pendulum compared to the other algorithms.

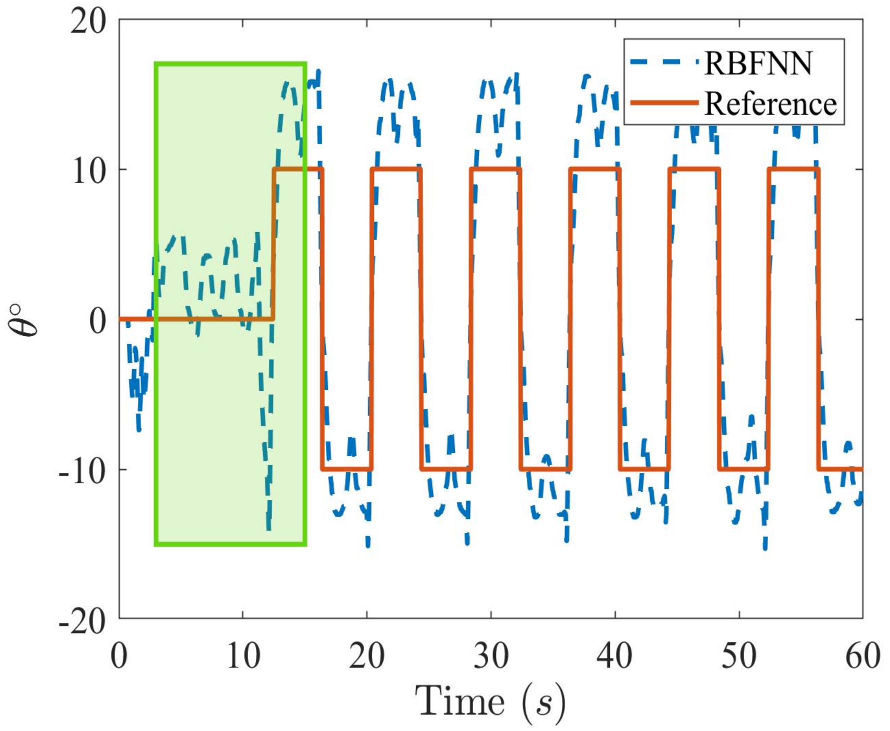


Figure 4.24: The trajectory tracking experimental data of Quanser-Servo2 with the pre-trained RBFNN optimal control and the selected data in the highlighted box for re-training.

The retrained RBFNN control is also applied to track a square wave. In this case, RL policies cannot track the desired trajectories. The tracking results of LQR, pre-trained RBFNN and retrained RBFNN are shown in Figure 4.27. It is shown that the retrained RBFNN control also shows much suppressed oscillations compared to the LQR and original RBFNN. A measure of control effort is defined in terms of the integrated absolute control voltage $v(t)$, which is given by,

$$u_{\text{effort}} = \sum_{i=1}^{n_s} |v(i)| \tag{4.59}$$

A summary of the total control effort and RMS error for balancing and trajectory tracking is listed Table 4.2.

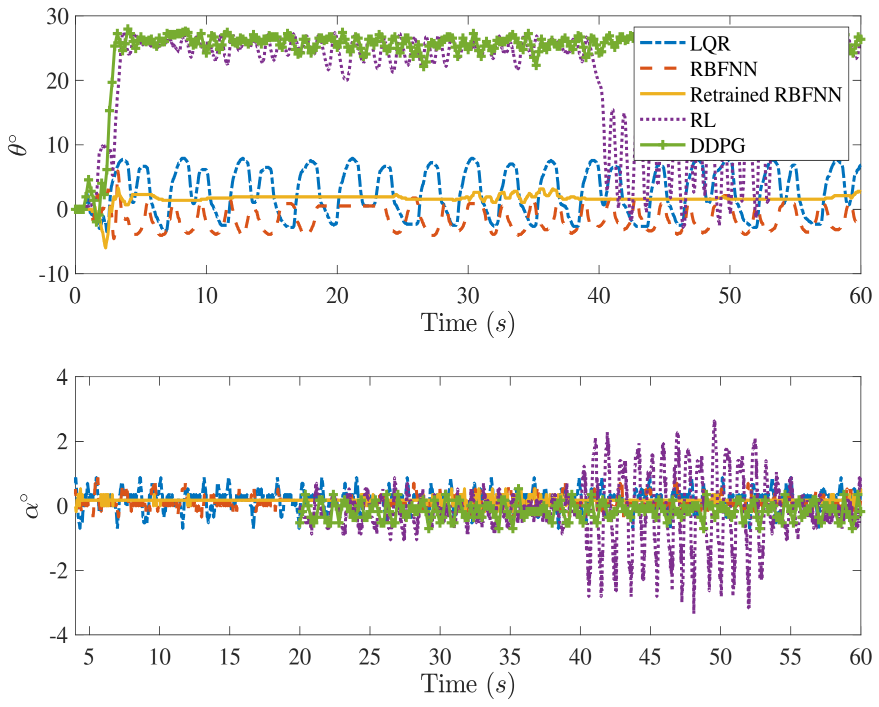


Figure 4.25: Rotation angle comparisons between LQR, pre-trained RBFNN, retrained RBFNN and the DDPG RL polices in terms of the balancing of Quanser-Servo2. Top: Angle of pendulum θ. Bottom: Angle of rotary arm α.

Table 4.2: Summary of control performance for LQR, RBFNN, retrained RBFNN and DDPG RL policies.

Case	Algorithm	RMS	u_{effort}
Stabilization	LQR	4.54	5610.35
	RBFNN	2.31	5238.32
	Retrained RBFNN	1.83	2132.15
	RL	4.99	3225.95
	DDPG	4.99	3087.94
Trajectory tracking	LQR	4.67	7410.10
	RBFNN	4.08	7468.42
	Retrained RBFNN	2.77	6667.54

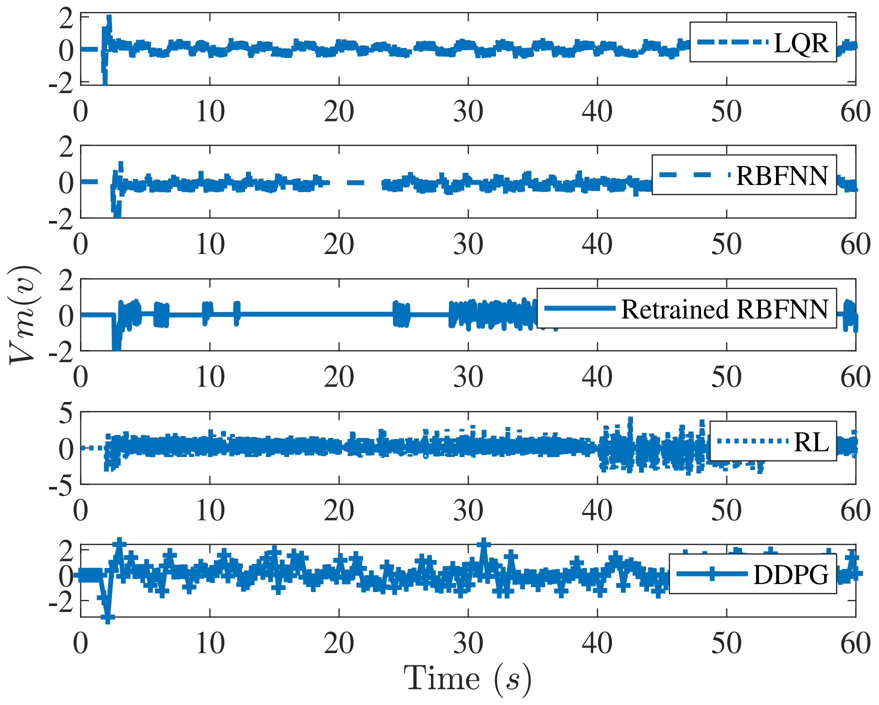


Figure 4.26: DC motor voltage comparisons between LQR, pre-trained RBFNN, retrained RBFNN and the DDPG RL polices in terms of the balancing of Quanser-Servo2.

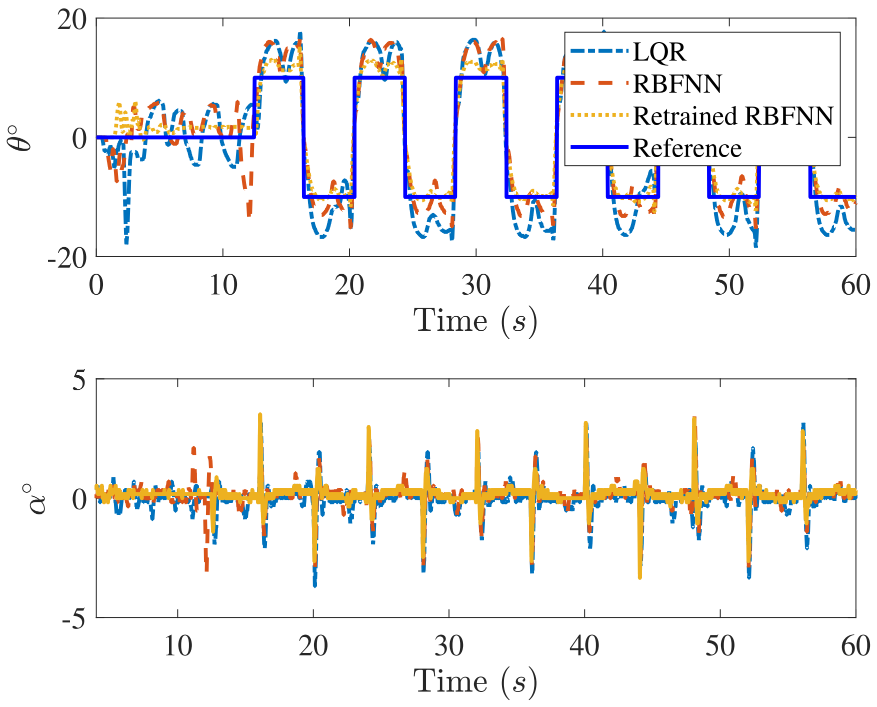


Figure 4.27: Rotation angle comparisons between LQR, pre-trained RBFNN and retrained RBFNN in terms of the trajectory tracking of Quanser-Servo2. Top: Angle of pendulum θ. Bottom: Angle of rotary arm α.

4.5.3 Qube2 with Empirical Balanced Truncation

In this section, the RBFNN optimal control is applied to the reduced order model from empirical balanced truncation. Both numerical simulations and experimetal validations are persented in this section to illustrate the performance of RBFNN optimal control with empirical balanced truncation.

4.5.3.1 Numerical Simulations

Figure 4.28 shows the relative output error e_m of the two reduced order models compared to the output of the original system when the same input $u_t(t) = e^{-100(t-0.2)^2}$ is applied. e_m is defined as

$$e_m = \sqrt{\frac{\sum_{t=1}^{n_s} |\mathbf{y}(t) - \mathbf{y}_r(t)|^2}{\sum_{t=1}^{n_s} |\mathbf{y}(t)|^2}} \tag{4.60}$$

where n_s is the number of integration steps and i is the discrete time index, corresponding to the physical time $t = i\Delta t$. Δt is the integration time step or sample time in experiments. The relative output error of the empirical balanced truncation is less than the model-based balanced truncation as seen in Figure 4.28. This indicates that the empirical gramians contain more accurate information than the model-based balanced gramians.

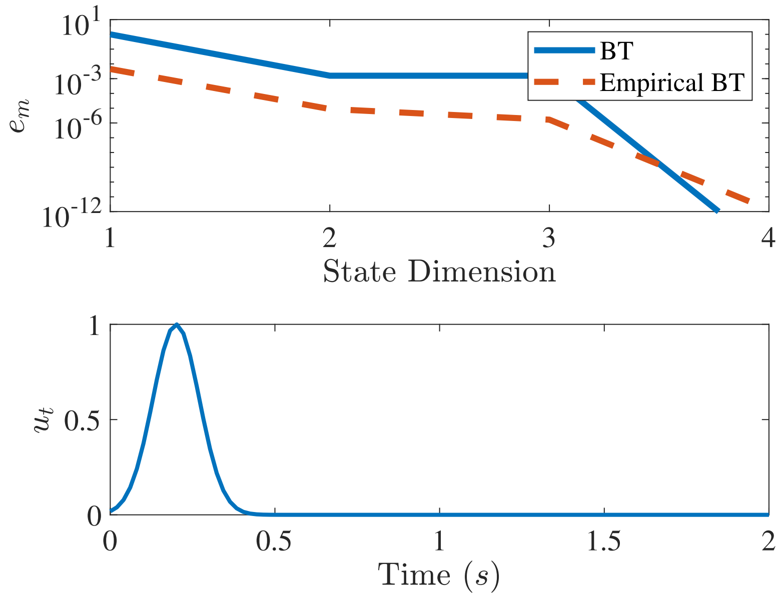


Figure 4.28: Top: Relative output errors of the reduced order model by the balanced truncation and empirical balanced truncation. Bottom: The input signal.

The LQR control of the original system is taken as a benchmark to compare with the performance of the RBFNN controls designed with the reduced order models. The **Q** and **R** matrices for the LQR control of the original system and the reduced order models are given by,

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q}_{bt} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.61}$$
$$\mathbf{Q}_{ebt} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 30 \end{bmatrix}, \ \mathbf{R} = \mathbf{R}_{bt} = \mathbf{R}_{ebt} = 1.$$

The sampling region for the RBFNN method with the reduced order model is chosen as $X_s = [-1, 1] \times [-1, 1] \times [-1, 1]$. The means of Gaussian neurons are uniformly distributed in the region: $X_g = [-4, 4] \times [-4, 4] \times [-4, 4]$. The standard deviations of Gaussian neurons are $\sigma_1 = \sigma_2 = \sigma_3 = 2.6667$. The number of neurons $N_G = 4^3 = 64$ and the sampling points $N_s = 2^3 N_G$. $\mathbf{w}_r$ denotes the trainable weights of the RBFNN for the reduced order model.

Consider tracking control again, the reference signal is chosen as,

$$\mathbf{x}_{ref} = [\theta_r, 0, 0, 0]^T \tag{4.62}$$

After that, the reference in the reduced order model space reads,

$$\mathbf{x}_{ref_r} = \mathbf{T}_r^{-1} \mathbf{x}_{ref} \tag{4.63}$$

The optimal control is given Equation (4.57). A root mean square (RMS) error is defined to quantify the performance of the tracking control,

$$e_{RMS} = \sqrt{\frac{1}{n_s} \sum_{i=1}^{n_s} (\theta(i) - \theta_r(i))^2} \tag{4.64}$$

We have chosen the time duration of 50 seconds and integration step $\Delta t = 0.001$ seconds. Hence, the number of integration steps $n_s = 50000$.

Consider a square wave reference. Figure 4.29 shows the performance of the controls under consideration. From the figure, we can see that the RBFNN optimal control has the ability to track the desired trajectory with different reduced order models and shows similar results compared to the LQR control of the original system. The RMS tracking errors of the RBFNN controls based on the model-based BT and empirical BT reduced order model and the LQR are 3.13, 3.16 and 3.71, respectively.

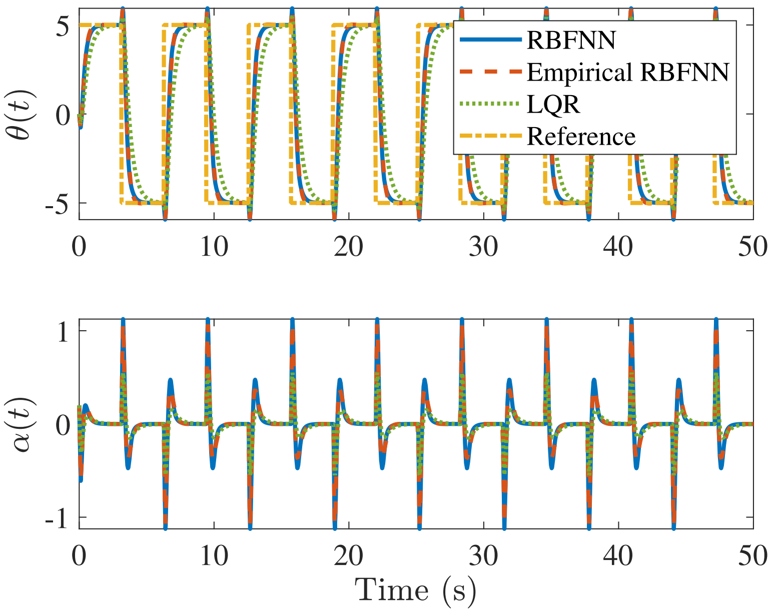


Figure 4.29: Tracking the square wave of the rotary army in simulations. In the legend, 'LQR' denotes the LQR control designed with the original model; 'RBFNN' denotes the RBFNN control designed with the model-based BT; 'Empirical RBFNN' denotes the RBFNN control designed with the empirical BT.

4.5.3.2 Experimental Validation

Two control experiments of pendulum balancing and rotary arm trajectory tracking are carried out on the Quanser-Servo2 Inverted Pendulum system with the RBFNN optimal controllers. All the experiments are done with Quanser HIL and MATLAB Simulink. The sample time is set to $\Delta t = 0.002$ seconds. The tests runs for 60 seconds. The closed-loop responses tracking the square wave as the reference signal for rotary arm are shown in Figure 4.30. Both the RBFNN optimal controls deliver better tracking performance than the LQR control.

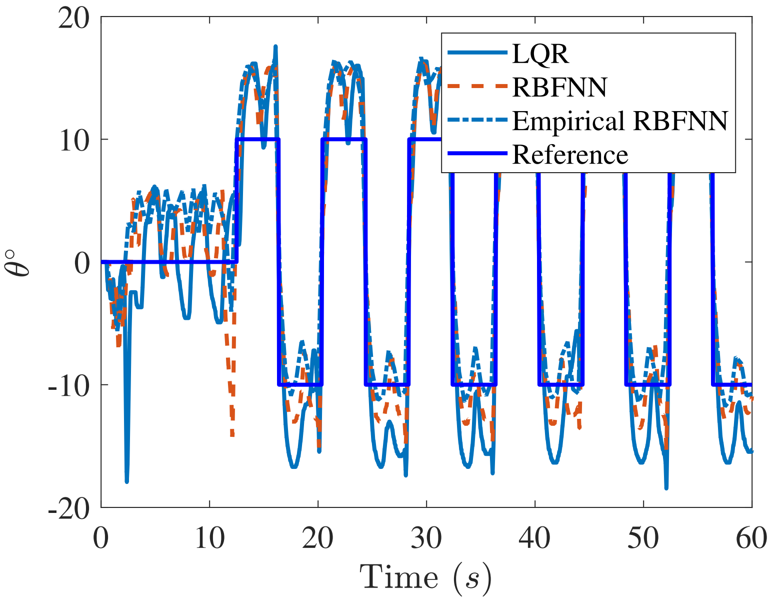


Figure 4.30: The closed-loop tracking responses $\theta(t)$ of the rotary arm of Quanser-Servo2. Legends are the same as in Figure 4.29.

Experimental data has been obtained to improve the control performance through retraining the neural networks. The transient responses of the system have been eliminated and the remaining data has been restricted to the time interval $t \in [3, 12]s$ for the model-based BT design and $t \in [3, 10]s$ for the empirical BT design. The effect of re-training on balancing control is shown in Figure 4.31. It is clear that the re-training of the neural networks with experimental data benefits the control designed with the empirical BT most.

Figure 4.32 shows the effect of re-training on tracking control. Compared to Figure 4.30, the performance improvement of tracking the angle $\theta(t)$ is obvious. Moreover, suppression of many spiky responses of $\alpha(t)$ angle in Figure 4.33 is a strong evidence of the benefits of re-training.

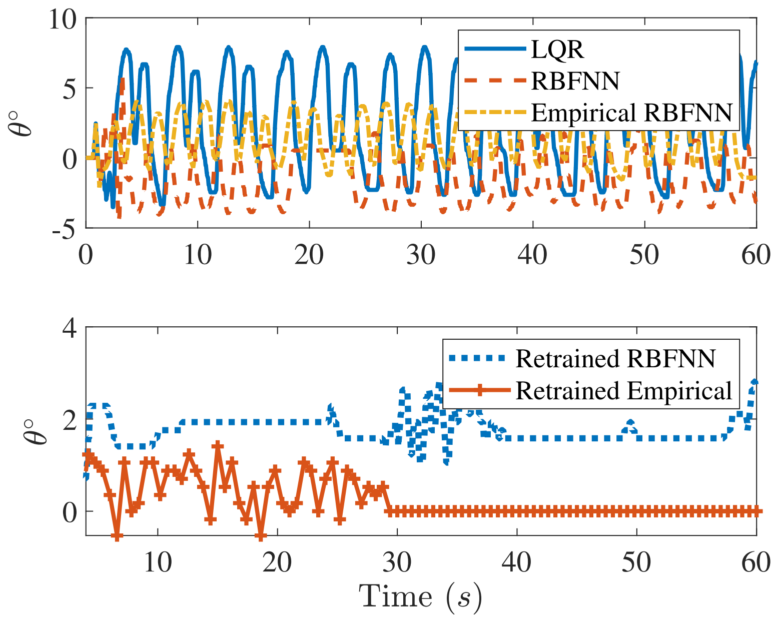


Figure 4.31: The closed-loop responses $\theta(t)$ of the rotary arm under various controls for balancing the inverted pendulum of Quanser-Servo2. Top: Responses before re-training. Bottom: Responses after re-training. Legends are the same as in Figure 4.29.

Table 4.3 lists the RMS tracking errors and total energies of all the controls under consideration including the effects of re-training. It is worth noting that the RBFNN control designed with empirical BT has the smallest RMS error.

Table 4.3: Summary of control performance for LQR, RBFNN, empirical RBFNN, retrained RBFNN and empirical retrained RBFNN.

Case	Algorithm	e_{RMS}	u_{effort}
Stabilization	LQR	4.54	4667.64
	RBFNN	2.31	4080.89
	Retrained RBFNN	1.83	2769.3
	Empirical RBFNN	2.86	4343.8
	Retrained Empirical	0.57	3210.1
Trajectory tracking	LQR	4.67	7410.1
	RBFNN	4.08	7468.4
	Retrained RBFNN	2.77	6667.5
	Empirical RBFNN	4.34	6794.8
	Retrained Empirical	3.21	8380.3

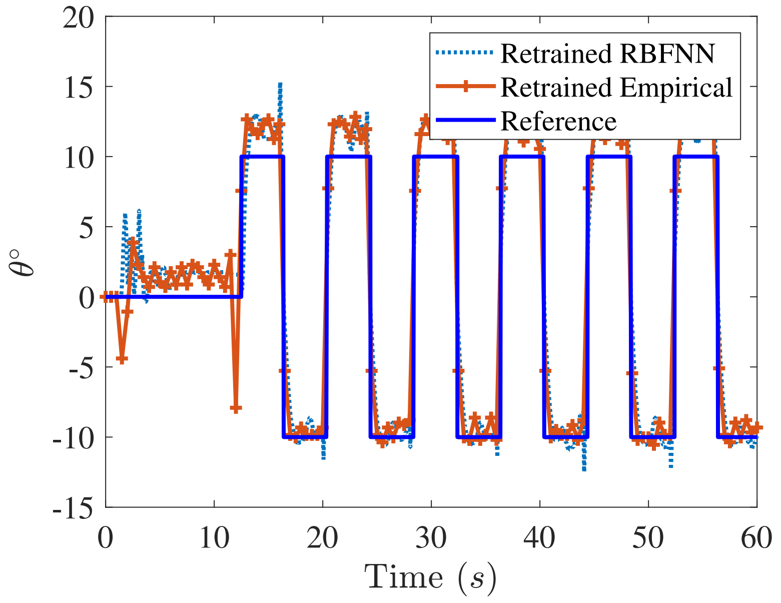


Figure 4.32: The closed-loop tracking response $\theta(t)$ of the rotary arm of Quanser-Servo2. Legends are the same as in Figure 4.29.

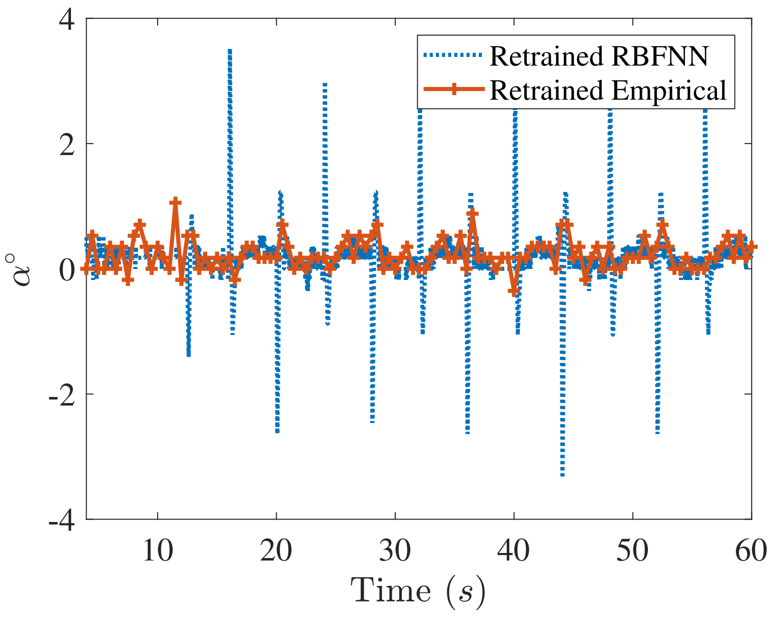


Figure 4.33: The closed-loop response $\alpha(t)$ of the pendulum in the rotary arm tracking control of Quanser-Servo2. Legends are the same as in Figure 4.29.

The robustness of the balancing control to disturbance was also investigated through experimental evaluation. A torque disturbance $d(t)$ is injected at $t \in [5, 5.5]s$ by introducing a voltage pulse to the motor. Figure 4.34 shows that the retrained empirical RBFNN has superb robustness to disturbance compared to the other two controls.

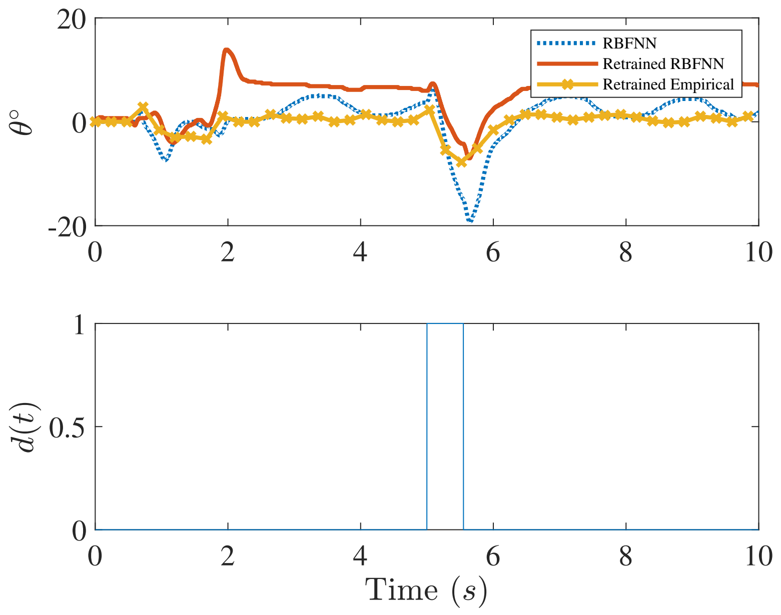


Figure 4.34: Robustness comparisons of all the controls under consideration. Top: The closed-loop angle response $\theta(t)$ of the rotary arm in balancing control of Quanser-Servo2. Bottom: Disturbance $d(t)$. Legends are the same as in Figure 4.29.

4.5.4 Conclusions

This section has applied the RBFNN optimal control approach with Gaussian activation functions for Quanser Qube2 system. Instead of analytically solving the HJB equation, the RBFNN compute the optimal control solution with high efficiency and performance. The optimal control solution is computed off-line in spatial domain for linear Qube2 system. Extensive numerical results and experimental studies have been presented and suggest that RBFNN optimal control with model reduction and transfer learning show good performance in terms of stabilization and robustness. The most meaningful advantage of RBFNN optimal control is that it shows significantly robustness improvement in terms of the optimal control for Qube2 system with reduced order model. Moreover, the RBFNN optimal control has the ability improve the performance by updating the neural networks weights through limited experimental data from Qube2.

4.6 Efficiency Improvement

The curse of dimensionality is a common issue in the neural networks solutions for solving the HJB equation. To improve the computation efficiency, we propose using RBFNN with separable Gaussian functions, taking advantage of the factorizable property of Gaussian functions. Moreover, it is found that the computation between the separable Gaussian function can be facilitated by the Kronecker product. In the following, a new nerual networks structure named separable Gaussian neural networks (SGNN) is introduced. Extensive numerical examples demonstrate that using SGNN in combination with the Kronecker product leads to significant speedups in computation efficiency, ranging from $3\times \sim 6\times$.

4.6.1 Separable Gaussian Neural Networks

The Gaussian functions are adopted as activation functions in the RBFNN, recall that the Gaussian function is defined as,

$$g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_{j,i}} \exp\left[-\frac{1}{2\sigma_{j,i}^2}(x_i - \mu_{j,i})^2\right]. \tag{4.65}$$

Equation (4.65) is equivalent to the product of multiple 1-D Gaussian functions expressed as,

$$g(\mathbf{x}, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = G_1(x_1) \cdot G_2(x_2) \cdots G_n(x_n) \tag{4.66}$$

where $G(x_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp(-\frac{(x_k-\mu_{jk})^2}{2\sigma_{jk}^2})$.

Here the separable Gaussian neural networks (SGNN) is proposed by making use of the factorization property of the Gaussian functions. The outputs of each hidden layer and output layer for SGNN are proposed as,

$$\mathbf{H}_1 = \mathbf{G}_1(\mathbf{x}_1, \boldsymbol{\mu}_1, \boldsymbol{\sigma}_1) \tag{4.67}$$

$$\mathbf{H}_2 = \mathbf{G}_2(\mathbf{x}_2, \boldsymbol{\mu}_2, \boldsymbol{\sigma}_2)(\sum_{i=1}^{ng_1}\sum_{j=1}^{ng_2} w_{ij}^1 h_{1i} + b_i) \tag{4.68}$$

$$\cdots \tag{4.69}$$

$$\mathbf{H}_n = \mathbf{G}_n(\mathbf{x}_n, \boldsymbol{\mu}_n, \boldsymbol{\sigma}_n)(\sum_{i=1}^{ng_{n-1}}\sum_{j=1}^{ng_n} w_{ij}^{n-1} h_{(n-1)i} + b_i) \tag{4.70}$$

$$\tag{4.71}$$

where $\mathbf{H}_n$ represents the output from *nth* hidden layer, w_{ij}^n represents the weights from *nth* layer, i and j represent the index of outputs from previous hidden layer and the current hidden layer. ng_n represents the number of neurons at the *nth* hidden layer, h_{ni} represents outputs i from Gaussian function at the nth hidden layer. Details of SGNN are given as:

1. Each Gaussian function can be considered as a single hidden layer in a feed-forward neural network. Each hidden layer are connected by weights w_{ij} and biases b_i.

2. Each state space serves as a seperable input to each hidden layer.

3. To slove HJB, the state space is discritized in the specified domain. In this case, the output from each hidden layer can be calculated by Kronecker product. Equation 4.66 can be written as,

$$g(\mathbf{x}) = \mathbf{G}_1(\mathbf{x}_1) \bigotimes \mathbf{G}_2(\mathbf{x}_2) \bigotimes \mathbf{G}_3(\mathbf{x}_3) \cdots \mathbf{G}_n(\mathbf{x}_n) \tag{4.72}$$

4. The weights w_{ij} and biases b_i between each hidden layer are set to be 1 and untrainable when solving the HJB equation.

5. For general applications, the standard deviations $\boldsymbol{\sigma}$, mean values $\boldsymbol{\mu}$, weights w_{ij} and biases b_i are all trainable.

A diagram can be found in th Figure 4.35 to illustrate the structure of separable Gaussian neural networks.

4.6.2 Numerical Simulations

The SGNN is found to be efficient to solve the HJB equations in optimal control. It is numerically computational expensive to solve the HJB equations when the dimension of the system $n \geq 4$. This issue can also be observed in the computation of neural networks. To improve the computation efficiency, the SGNN is adopted to solve the optimal control problem in terms of different dynamic systems. Additional, the control law $\mathbf{u}$ and the drivative of the Gaussian functions $\mathbf{D}_g$ are computated on each dimension seperately to improve the calculation efficiency. The algorithm is shown in Algorithm 1. The matrices in Algorithm 1 are defined following the RBFNN optimal control algorithm in Section 4.2.

Table 4.4 summarizes the performance of SGNN and compares it with RBFNN, using 2D, 3D, and 4D systems as examples. Here N_G denotes the total number of neurons, N_S denotes the total number of sampling points. Details of the design for 2D, 3D and 4D systems can be found from Section 4.6.2.2 to Section 4.6.2.3. Figure 4.36 shows the computation time comparisons for different cases with RBFNN and SGNN. From Figure 4.36 it can be found that compared to RBFNN, the achieved speedups of SGNN are in the range of $3\times \sim 6\times$.

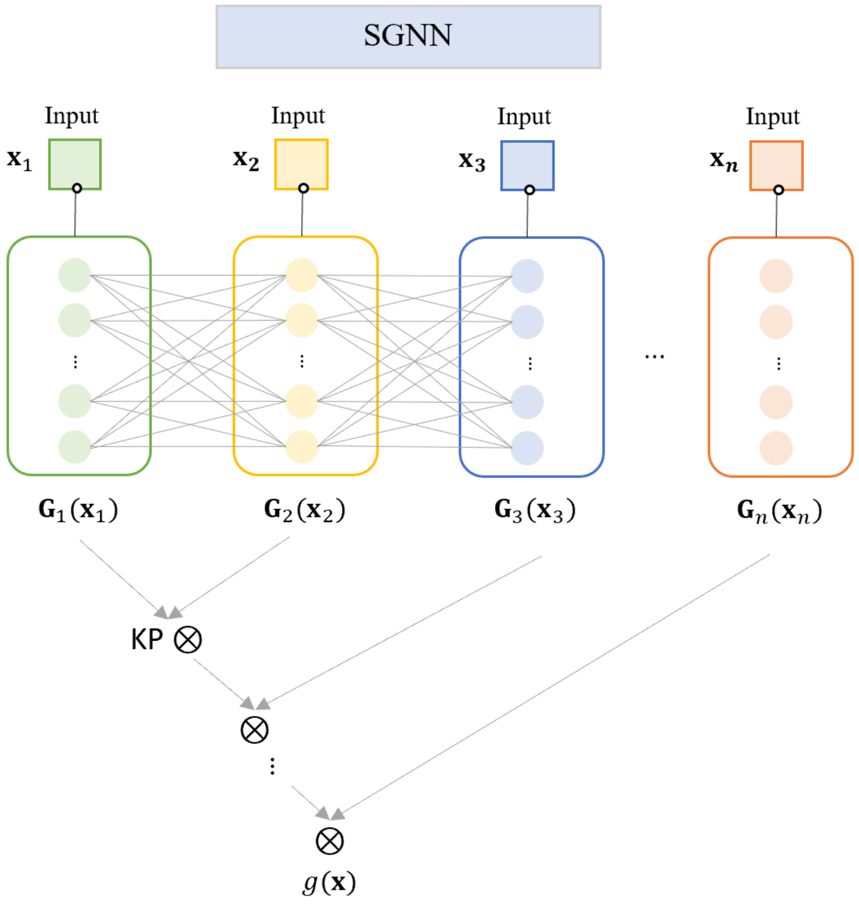

SGNN
Input
Input
Input
Input
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_n$
$\mathbf{G}_1(\mathbf{x}_1)$
$\mathbf{G}_2(\mathbf{x}_2)$
$\mathbf{G}_3(\mathbf{x}_3)$
$\mathbf{G}_n(\mathbf{x}_n)$
KP $\otimes$
$g(\mathbf{x})$

Figure 4.35: Flowchart of the separable Gaussian neural networks.

4.6.2.1 Case 1: 2D System

Consider the linear 2D system in Section 4.3.3.1,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{4.73}$$

where $m = 1$, $c = 2$ and $k = 1$. The neural networks region is chosen as $X_g \in [-2, 2] \times [-6, 6]$ and sampling region is chosen as $X_s \in [-1, 1] \times [-3, 3]$. The number of neurons $N_G \times N_G$ are 12×12 and the sampling points are $(2N_G - 1) \times (2N_G - 1)$ along the state space vectors. The standard deviations are chosen to be $\sigma_1 = 0.4444$, $\sigma_2 = 1.3333$.

4.6.2.2 Case 2: 3D System

Consider the linear 3D system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -2 & -1 & 1 \\ 1 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 & u_2 \end{bmatrix} \tag{4.74}$$

The neural networks region is chosen as $X_g \in [-3,3]\times[-3,3]\times[-3,3]$ and sampling region is chosen as $X_s \in [-2,2]\times[-2,2]\times[-2,2]$. The number of neurons $N_G \times N_G$ are $10 \times 10 \times 10$ and the sampling points are $13 \times 13 \times 13$ along the state space vectors. The standard deviations are chosen to be $\sigma_1 = \sigma_2 = \sigma_3 = 0.6667$.

4.6.2.3 Case 3: 4D System

Consider the linear 4D system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} -2 & -1 & 1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} u \tag{4.75}$$

The neural networks region is chosen as $X_g \in [-3,3]\times[-3,3]\times[-3,3]\times[-3,3]$ and sampling region is chosen as $X_s \in [-1,1]\times[-1,1]\times[-1,1]\times[-1,1]$. The number of neurons $N_{G1} \times N_{G2} \times N_{G3} \times N_{G4}$ are $5 \times 5 \times 5 \times 5$ and the sampling points are $7 \times 7 \times 7 \times 7$ along the state space vectors. The standard deviations are chosen to be $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 1.5$.

Table 4.4: Summary of computation performance for RBFNN and SGNN.

Neural Networks	Performance	2D	3D	4D
	Time(s)	0.7789	4.2669	13.3796
SGNN	N_G	144	1000	625
	N_S	529	1331	2401
	Time(s)	2.9278	23.0905	56.8583
RBFNN	N_G	144	1000	625
	N_S	529	1331	2401

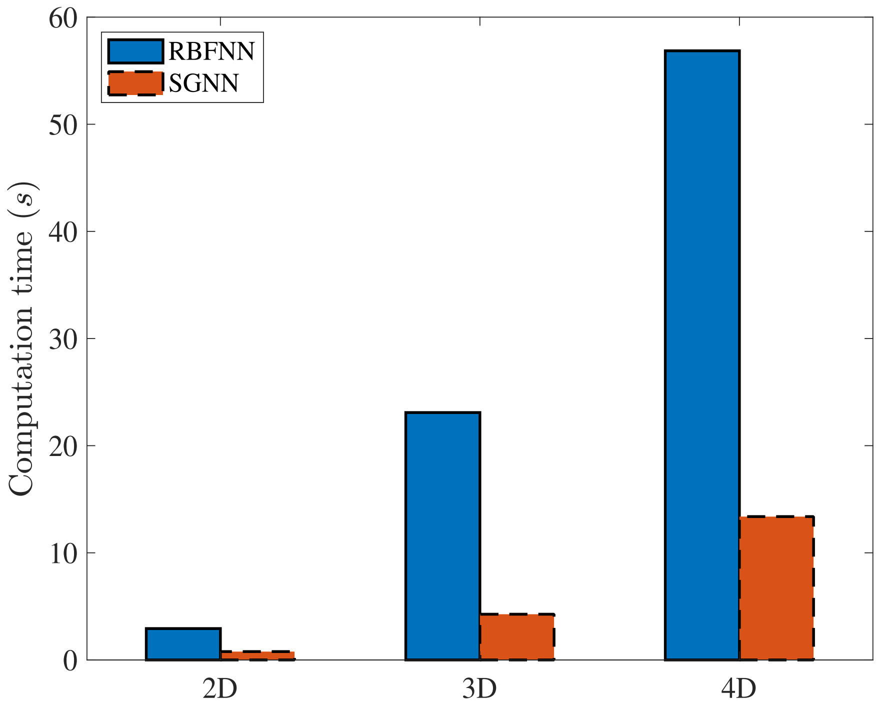


Figure 4.36: Computation time comparisons between RBFNN and SGNN for 2D, 3D and 4D systems.

Algorithm 1 SGNN solution for optimal control algorithm

```
1: **RBF setting**: Number of neurons $N_G$, number of sampling points$N_S$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are fixed.
2: **LQR setting**: $\mathbf{Q}$, $\mathbf{R}$ are diagonal matrices.
3: **Initialize**:
4: $\mathbf{w}_0 \leftarrow N_G \times 1$    ▷ Initialize the initial weights of RBF
5: $\mathbf{u} \leftarrow N_S \times m$    ▷ m is the number of controls.
6: $k \leftarrow 1$    ▷ Initialize iteration step
7: **while** i $< N_s$ **do**
8:     $\mathbf{D}_g(i,:,:) \leftarrow N_S \times N_G \times n$    ▷ Calculate the matrix $\mathbf{D}_g$ from Kronecker product, n is the dimension of the system.
9:     $\mathbf{S}_1 = \mathbf{f}(\mathbf{x}_s(:,i)) * \mathbf{D}_g(i,:,:)$
10:    $\mathcal{L}_1 = \frac{1}{2} * \mathbf{x}_s(:,i)^T * \mathbf{Q} * \mathbf{x}_s(:,i)$
11:    $i = i + 1$
12: **end while**
13: $\mathbf{g}_t = \mathbf{g}^T \leftarrow m \times n$
14: **while** k $\leq$ Time points **do**
15:     $\mathbf{S}_2 \leftarrow N_S \times N_G$    ▷ All the elements in $\mathbf{S}_2$ are set to be zero
16:     $\mathcal{L}_2 \leftarrow N_S \times 1$    ▷ All the elements in $\mathcal{L}_2$ are set to be zero
17:     $\mathbf{BD}_g \leftarrow N_S \times m$    ▷ All the elements in $\mathbf{BD}_g$ are set to be zero
18:     **while** i $<$ m **do**
19:         **while** j $<$ n **do**
20:             $\mathbf{BD}g(:,i) = \mathbf{BD}g(:,i) + \mathbf{g}_t(i,j) * \mathbf{D}_g(:,:,j) * \mathbf{w}(:,k)$
21:             $j = j + 1$
22:         **end while**
23:         $\mathbf{u}(:,i) = -1/\mathbf{R}(i,i) * \mathbf{BD}_g(:,i)$
24:         $\mathcal{L}_2 = \mathcal{L}_2 + \frac{1}{2} * \mathbf{u}(:,i) \cdot \mathbf{u}(:,i) * \mathbf{R}(i,i)$    ▷ $\cdot$ represent the doc product
25:         $i = i + 1$
26:     **end while**
27:     $\mathbf{uTBT} = \mathbf{u} * \mathbf{g}_t \leftarrow N_s \times n$
28:     **while** i $<$ n **do**
29:         $\mathbf{S}_2 = \mathbf{S}_2 + \mathbf{uTBT}(:,i) \cdot \mathbf{D}_g(:,:,i)$
30:         $i = i + 1$
31:     **end while**
32:     $\mathbf{S} = \mathbf{S}_1 + \mathbf{S}_2$
33:     $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$
34:     $\mathbf{A}_g = \mathbf{S}^T * \mathbf{S}$
35:     $\mathbf{b} = \mathcal{L} * \mathbf{S}$
36:     $\mathbf{w}(:,k+1) = -\mathbf{A}_g^{-1} * \mathbf{b}$
37:     $k \leftarrow k + 1$
38: **end while**
```

4.7 Conclusions

This section has presented a RBFNN optimal control approach with Gaussian activation functions. Instead of analytically solving the HJB equation, the RBFNN compute the optimal control solution with high efficiency and performance. The optimal control solution is computed off-line in the spatial domain for linear dynamic systems. Extensive numerical results and experimental studies have been presented and suggest that the RBFNN optimal control with model reduction and transfer learning show good performance in terms of computational cost, better approximation and control performance owing to its simple and fixed architecture. Optimal control from LQR and neural networks with reinforcement learning are chosen as benchmarks to compare with the performance of the proposed RBFNN control. The significant advantage of RBFNN optimal control is that it can improve the control performance by updating the neural networks weights through limited experimental data. Balanced truncation is introduced to reduce the order of the dynamic systems so as to improve the computation efficiency. Moreover, a seperable Gaussian neural networks is first proposed to solve the optimal control problem and improve the computation process of the neural networks. Numerical examples show that SGNN speeds up to $3\times \sim 6\times$ compared to original RBFNN.

Chapter 5

EXTENDED STATE OBSERVERS WITH RECURSIVE LEAST SQUARES ALGORITHM

The ESO was first proposed in [89] and is one of the core concepts in active disturbance rejection control (ADRC). A linear ESO was proposed in [90], which was much simpler to design than the general nonlinear ESO. Without exploiting the mathematical model of uncertainties, the ESO has been proven to be an effective approach to estimate disturbances and uncertainties [91]. This is the key advantage of the ESO. The ESO-based controls have been widely studied including the control of DC motor [92], wind energy system [93] and hypersonic vehicles [94]. A good survey about the development of the extended state observer can be found in reference [95]. The extensive applications and accurate performance of the ESO motivate us to explore it for estimation of unknown dynamics. Without dealing with either the complex matrix calculation in the Kalman filter or the data requirements in machine learning, the ESO is simple to implement and yields the same accurate results as the Kalman filter and neural networks would. To implement the ESO, the reconstructed extended observer system is required to be observable. The observability of the ESO has drawn much attention from researchers when dealing with large scale systems. The sufficient condition for observability of the ESO for structural systems with rigid body and elastic modes has been found in [96]. To investigate the observability of the ESO for reconstructed extended observer system, the special structure of the ESO helps us find the minimum number of sensors to estimate the unkonwn dynamics, which has not been studied in other publications to the best of the author's knowledge.

5.1 Extended State Observer

Consider the LTI system with unknown dynamics given as,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}(t) + \mathbf{B}_f\mathbf{f}(t), \mathbf{x}(t_0) = \mathbf{x}_0 \tag{5.1}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{5.2}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times q}$, $\mathbf{B}_f \in \mathbb{R}^{n\times p}$, $\mathbf{f}(t) \in \mathbb{R}^{p\times 1}$, $\mathbf{C} \in \mathbb{R}^{m\times n}$. In this section, we design the observers to estimate the time-varying unknown dynamics $\mathbf{f}(t)$. In order to estimate the unknown dynamics $\mathbf{f}(t)$, we introduce a new state as,

$$\mathbf{x}_n(t) = \mathbf{f}(t) \in \mathbb{R}^{p\times 1} \tag{5.3}$$

Define an extended state vector as $\mathbf{x}_e(t) = [\mathbf{x}(t); \mathbf{x}_n(t)] \in \mathbb{R}^{(n+p)\times 1}$. The extended state equation reads,

$$\begin{aligned} \dot{\mathbf{x}}_e(t) &= \mathbf{A}_e\mathbf{x}_e(t) + \mathbf{B}_e\mathbf{u}(t) + \mathbf{B}_h\mathbf{h} \\ \mathbf{y}_e(t) &= \mathbf{C}_e\mathbf{x}_e(t) \end{aligned} \tag{5.4}$$

where the output of the extended state system is the same as the original output $\mathbf{y}_e(t) = \mathbf{y}(t) \in \mathbb{R}^{m\times 1}$. The extended state system matrices and vectors $\mathbf{A}_e$, $\mathbf{B}_e$, $\mathbf{C}_e$, $\mathbf{B}_h$, and $\mathbf{h}$ are given by

$$\begin{aligned} \mathbf{A}_e &= \begin{bmatrix} \mathbf{A}_s & \mathbf{B}_f \\ \mathbf{0}_{p\times n} & \mathbf{0}_{p\times p} \end{bmatrix} \in \mathbb{R}^{(n+p)\times(n+p)}, \quad \mathbf{B}_e = \begin{bmatrix} \mathbf{B}_u \\ \mathbf{0}_{p\times q} \end{bmatrix} \in \mathbb{R}^{(n+p)\times q} \\ \mathbf{C}_e &= \left[\mathbf{C}, \mathbf{0}_{m\times p}\right] \in \mathbb{R}^{m\times(n+p)}, \quad \mathbf{B}_h = \begin{bmatrix} \mathbf{0}_{n\times p} \\ \mathbf{I}_p \end{bmatrix} \in \mathbb{R}^{(n+p)\times p} \\ \mathbf{h} &\approx \dot{\mathbf{f}}(t) \in \mathbb{R}^{p\times 1} \end{aligned} \tag{5.5}$$

It should be noted that $\mathbf{h}$ is an estimate of the derivative of the unknown dynamics $\dot{\mathbf{f}}(t)$. More discussions about it will be presented later. A linear extended state observer (LESO) for the extended state system in Equation (5.4) can be designed as [90]

$$\dot{\hat{\mathbf{x}}}_e(t) = \mathbf{A}_e\hat{\mathbf{x}}_e(t) + \mathbf{B}_e\mathbf{u}(t) + \mathbf{L}(\mathbf{y}(t) - \hat{\mathbf{y}}(t)) \tag{5.6}$$

$$\hat{\mathbf{y}}(t) = \mathbf{C}_e\hat{\mathbf{x}}_e(t) \tag{5.7}$$

where $\mathbf{L}$ is the observer gain matrix, $\hat{\mathbf{x}}_e(t)$ is an estimate of the extended state $\mathbf{x}_e(t)$. Define the estimation error as $\tilde{\mathbf{x}}_e(t) = \mathbf{x}_e(t) - \hat{\mathbf{x}}_e(t)$. Making use of Equation (5.6), we obtain

$$\dot{\tilde{\mathbf{x}}}_e(t) = (\mathbf{A}_e - \mathbf{L}\mathbf{C}_e)\tilde{\mathbf{x}}_e(t) + \mathbf{B}_h\mathbf{h} \tag{5.8}$$

Since the term $\mathbf{B}_h\mathbf{h}$ can be viewed as an external disturbance to the observer, when the pair $(\mathbf{A}_e, \mathbf{C}_e)$ is observable, we can find a gain matrix $\mathbf{L}$ to make the matrix $\mathbf{A}_e - \mathbf{L}\mathbf{C}_e$ Hurwitz stable. Hence, when the ESO system is observable and $\mathbf{h}$ is bounded, the estimation error $\tilde{\mathbf{x}}_e(t)$ will be bounded.

The sensor system design usually addresses these issues: what and how many of physical quantities should be measured as the outputs so that the ESO in Equation (5.6) is observable. In the following sections, we first study the observability of the

ESO and then the sensor choices to estimate the unknown dynamics in the system.

5.2 Observability Analysis

The observability matrix is defined as

$$\mathcal{O} = \begin{bmatrix} \mathbf{C}_e \\ \mathbf{C}_e\mathbf{A}_e \\ \mathbf{C}_e\mathbf{A}_e^2 \\ \vdots \\ \mathbf{C}_e\mathbf{A}_e^{n-1} \end{bmatrix} \tag{5.9}$$

When $\mathcal{O}$ has a full rank n, the system is observable. However, when the order of the system is high ($n \gg 1$) and when the state matrix $\mathbf{A}_e$ is ill-conditioned, the numerical evaluation of the rank of $\mathcal{O}$ becomes highly unreliable.

An alternative way to check observability is to use the PBH test as stated in the following theorem.

Theorem 5.2.1. *Consider the extended state matrix $\mathbf{A}_e \in \mathbb{R}^{(n+p)\times(n+p)}$ and the output matrix $\mathbf{C}_e \in \mathbb{R}^{m\times(n+p)}$. The pair ($\mathbf{A}_e$,$\mathbf{C}_e$) is observable if and only if the $(n+p+m)\times(n+p)$ matrix,*

$$\mathcal{O}_\lambda \equiv \begin{bmatrix} \mathbf{A}_e - \lambda_e \mathbf{I}_{(n+p)} \\ \mathbf{C}_e \end{bmatrix} \tag{5.10}$$

has rank $n+p$ for every eigenvalue λ_e of $\mathbf{A}_e$, and for the corresponding right eigenvector $\mathbf{a}$ of $\mathbf{A}_e$ such that $\mathbf{C}_e\mathbf{a} \neq \mathbf{0}$.

The proof of the theorem can be found in [97]. We shall apply Theorem 5.2.1 to determine the observability of the pair ($\mathbf{A}_e$, $\mathbf{C}_e$) for the ESO.

5.2.1 Properties of Block Matrix Rank

Let us first review the mathematical properties of the rank of block matrices. Consider the matrices $\mathbf{M} \in \mathbb{R}^{m\times n}$, $\mathbf{N} \in \mathbb{R}^{l\times n}$, $\mathbf{P} \in \mathbb{R}^{m\times k}$, and $\mathbf{0} \in \mathbb{R}^{l\times k}$. We use $r(\cdot)$ to denote the rank of a matrix. The rank of block matrices has the following equality properties [98, 99].

1.
$$r\left(\begin{bmatrix} \mathbf{M} & \mathbf{P} \\ \mathbf{N} & \mathbf{0} \end{bmatrix}\right) = r(\mathbf{N}) + r(\mathbf{P}) + r((\mathbf{I}_m - \mathbf{P}\mathbf{P}^+)\mathbf{M}(\mathbf{I}_n - \mathbf{N}^+\mathbf{N})) \tag{5.11}$$

2.

$$r(\mathbf{I}_n - \mathbf{M}^+\mathbf{M}) = n - r(\mathbf{M}) \tag{5.12}$$
$$r(\mathbf{I}_m - \mathbf{M}\mathbf{M}^+) = m - r(\mathbf{M}) \tag{5.13}$$

where $\mathbf{M}^+$, $\mathbf{N}^+$, $\mathbf{P}^+$ are defined as the Moore-Penrose inverses of the matrix $\mathbf{M}$, $\mathbf{N}$ and $\mathbf{P}$.

5.2.2 Rank of Observability Matrix $\mathcal{O}_\lambda$

In this section, we shall find the necessary condition for observability of the ESO. Before we present a lemma, let us introduce the matrices involved in the observability.

The matrices for the ESO are given by,

$$\mathbf{A}_e = \begin{bmatrix} \mathbf{A}_s & \mathbf{B}_f \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(n+p)\times(n+p)},\ \mathbf{C}_e = \begin{bmatrix} \mathbf{C}, \mathbf{0} \end{bmatrix} \in \mathbb{R}^{m\times(n+p)} \tag{5.14}$$

where $\mathbf{A}_s \in \mathbb{R}^{n\times n}$, $\mathbf{B}_f \in \mathbb{R}^{n\times p}$, $\mathbf{C} \in \mathbb{R}^{m\times n}$. λ_e is an eigenvalue of matrix $\mathbf{A}_e$. We apply the PBH test in Theorem 5.2.1 to determine the observability of pair $(\mathbf{A}_e, \mathbf{C}_e)$ and study the rank of the matrix $\mathcal{O}_\lambda$ for every eigenvalue λ_e of the matrix $\mathbf{A}_e$,

$$\mathcal{O}_\lambda = \begin{bmatrix} \mathbf{A}_e - \lambda_e \mathbf{I}_{(n+p)} \\ \mathbf{C}_e \end{bmatrix} = \begin{bmatrix} \mathbf{A}_s - \lambda_e \mathbf{I}_n & \mathbf{B}_f \\ \mathbf{0} & -\lambda_e \mathbf{I}_p \\ \mathbf{C} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{sh} & \mathbf{B}_{fh} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \tag{5.15}$$

where $\mathbf{B}_{fh}$ and $\mathbf{A}_{sh}$ are defined as,

$$\mathbf{A}_{sh} = \begin{bmatrix} \mathbf{A}_s - \lambda_e \mathbf{I}_n \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(n+p)\times n}, \mathbf{B}_{fh} = \begin{bmatrix} \mathbf{B}_f \\ -\lambda_e \mathbf{I}_p \end{bmatrix} \in \mathbb{R}^{(n+p)\times p} \tag{5.16}$$

Lemma 5.2.2. *The observability matrix $\mathcal{O}_\lambda$ is full rank if the following conditions hold.*

1.

$$\begin{aligned} r((\mathbf{I}_n - \mathbf{B}_{fh}\mathbf{B}_{fh}^+)\mathbf{A}_{sh}(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) &= \min(r(\mathbf{I}_n - \mathbf{B}_{fh}\mathbf{B}_{fh}^+), r(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \quad \text{for } \lambda_e \neq 0 \\ r((\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+)\mathbf{A}_s(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) &= \min(r(\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+), r(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \quad \text{for } \lambda_e = 0 \end{aligned} \tag{5.17}$$

2. The number of the output $m \geq p$

where m is the number of rows of the matrix $\mathbf{C}$, p is the number of columns of the matrix $\mathbf{B}_f$ and λ_e is an eigenvalue of the matrix $\mathbf{A}_e$.

Proof. Making use of Property 1 of block matrices in Equation (5.11), we obtain the rank of the matrix $\mathcal{O}_\lambda$ as,

$$\begin{aligned} r(\mathcal{O}_\lambda) &= r(\mathbf{C}) + r(\mathbf{B}_{fh}) + r((\mathbf{I}_{n+p} - \mathbf{B}_{fh}\mathbf{B}_{fh}^+)\mathbf{A}_{sh}(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \\ &= m + p + r((\mathbf{I}_{n+p} - \mathbf{B}_{fh}\mathbf{B}_{fh}^+)\mathbf{A}_{sh}(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \end{aligned} \tag{5.18}$$

where we have used the results $r(\mathbf{C}) = m$ and $r(\mathbf{B}_{fh}) = r(\mathbf{B}_f) = p$.

Since the eigenvalues of the matrix $\mathbf{A}_e$ consist of the non-zero eigenvalues of the matrix $\mathbf{A}_s$ and p zero eigenvalues associated with the extended states. We consider these two cases separately.

Making use of Property 2, we have

$$r(\mathbf{I}_{n+p} - \mathbf{B}_{fh}\mathbf{B}_{fh}^+) = n + p - r(\mathbf{B}_{fh}) = n \tag{5.19}$$

$$r(\mathbf{I}_n - \mathbf{C}^+\mathbf{C}) = n - r(\mathbf{C}) = n - m \tag{5.20}$$

If Condition 1 in Lemma 5.2.2 holds, we have

$$\begin{aligned} &r((\mathbf{I}_{n+p} - \mathbf{B}_{fh}\mathbf{B}_{fh}^+)\mathbf{A}_{sh}(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \\ &= \min(r(\mathbf{I}_{n+p} - \mathbf{B}_{fh}\mathbf{B}_{fh}^+), r(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) = n - m \end{aligned} \tag{5.21}$$

Hence,

$$r(\mathcal{O}_\lambda) = n + p \tag{5.22}$$

That is, the observability matrix $\mathcal{O}_\lambda$ is full rank.

For the zero eigenvalues $\lambda_e = 0$, the rank of the block matrix can be obtained as follows.

$$\begin{aligned} r(\mathcal{O}_\lambda) &= r\left(\begin{bmatrix} \mathbf{A}_s & \mathbf{B}_f \\ \mathbf{0} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} \end{bmatrix}\right) = r\left(\begin{bmatrix} \mathbf{A}_s & \mathbf{B}_f \\ \mathbf{C} & \mathbf{0} \end{bmatrix}\right) \\ &= r(\mathbf{C}) + r(\mathbf{B_f}) + r((\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+)\mathbf{A}_s(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \end{aligned} \tag{5.23}$$

Based on Property 2, the rank of matrices $(\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+)$ can be obtained as,

$$r(\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+) = n - r(\mathbf{B}_f) = n - p \tag{5.24}$$

When both Conditions 1 and 2 ($m \geq p$) of Lemma 5.2.2 hold, we obtain the following,

$$\begin{aligned} &r((\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+)\mathbf{A}_s(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) \\ &= \min(r(\mathbf{I}_n - \mathbf{B}_f\mathbf{B}_f^+), r(\mathbf{I}_n - \mathbf{C}^+\mathbf{C})) = n - m \end{aligned} \tag{5.25}$$

Hence, the matrix $\mathcal{O}_\lambda$ has a full rank, which is given by,

$$r(\mathcal{O}_\lambda) = m + p + (n - m) = n + p \tag{5.26}$$

□

Remark 5.2.1. We should point out the implication of the second condition, i.e. $m \geq p$, of Lemma 5.2.2. The number p of independent unknown dynamics terms is clearly a lower bound for the number of sensors needed in order to make the extended state observable.

Another condition of Theorem 5.2.1, $\mathbf{C}_e\mathbf{a} \neq \mathbf{0}$, has to be checked separately to confirm the observability of the extended state system.

5.3 Recursive Least Squares Algorithm

Once the unknown dynamics are estimated with the help of extended state observer, the recursive least squares (RLS) algorithm [100] can be applied to estimate the system parameters contained in the dynamics. The RLS algorithm is one of the least squares algorithms that is used to estimate the parameters of a linear model in real-time. The RLS algorithm is particularly useful when the data is collected sequentially over time, through computing the optimal estimate of the regression coefficients by minimizing the sum of the squared errors between the observations and predictions. One of the advantages of the RLS algorithm is that it can identify dynamic system parameters in real-time, saving computation power. The RLS algorithm is a powerful tool for modeling and predicting time-series data with wide applications to signal processing, control systems and communications. The specific details of how the algorithm is computed can be found below.

Define an objective function as,

$$J = \frac{1}{2}\sum_{i=1}^{n_s}(\hat{f}(i) - \boldsymbol{\varphi}^T(i)\hat{\boldsymbol{\eta}})^2 \tag{5.27}$$

where n_s is the number of sampled data points of the motion signals and $\hat{\boldsymbol{\eta}}$ is an estimate of the parameter vector $\boldsymbol{\eta} \in \mathbb{R}^{n_p \times 1}$. n_p is the number of parameters to estimate. $\hat{f}(i)$ is the estimated unknown dynamics at each sampling point, $\hat{\mathbf{f}} \in \mathbb{R}^{n_s \times 1}$. Note that same derivations can be extended to $\hat{\mathbf{f}} \in \mathbb{R}^{n_s \times p}$. $\hat{\boldsymbol{\eta}}$ is determined to minimize the objective function J. Introduce a data vector and a data matrix as

$$\hat{\mathbf{f}} = [\hat{f}(1), \hat{f}(2), \cdots, \hat{f}(n_s)]^T \in \mathbb{R}^{n_s \times 1} \tag{5.28}$$

$$\boldsymbol{\phi}(n_s) = [\boldsymbol{\varphi}(1), \boldsymbol{\varphi}(2), \cdots \boldsymbol{\varphi}(n_s)]^T \in \mathbb{R}^{n_s \times n_p} \tag{5.29}$$

Assume that for a given number n_s of data points, the data matrix $\boldsymbol{\phi}^T\boldsymbol{\phi}(n_s) \in \mathbb{R}^{n_p \times n_p}$ is non-singular. The optimal estimate of the parameter can be obtained as,

$$\hat{\boldsymbol{\eta}}(n_s) = \mathbf{P}(n_s)\boldsymbol{\phi}^T\hat{\mathbf{f}} \tag{5.30}$$

where

$$\mathbf{P}(n_s) = (\boldsymbol{\phi}^T(n_s)\boldsymbol{\phi}(n_s))^{-1} \tag{5.31}$$

The optimal estimate can be computed with the recursive least squares algorithm when there is a sufficient number of data points, i.e. when $n_s \gg 1$ [100]. Making use of Equations (5.30) and (5.31), we re-write the estimated parameter as,

$$\begin{aligned}
\hat{\boldsymbol{\eta}}(n_s) &= \mathbf{P}(n_s)\sum_{i=1}^{n_s}\boldsymbol{\varphi}(i)\hat{f}(i) &(5.32)\\
&= \mathbf{P}(n_s)\left[\sum_{i=1}^{n_s-1}\boldsymbol{\varphi}(i)\hat{f}(i) + \boldsymbol{\varphi}(n_s)\hat{f}(n_s)\right] &(5.33)\\
&= \mathbf{P}(n_s)[\mathbf{P}^{-1}(n_s-1)\hat{\boldsymbol{\eta}}(n_s-1) + \boldsymbol{\varphi}(n_s)\hat{f}(n_s)] &(5.34)
\end{aligned}$$

From Equation (5.31), we have

$$\begin{aligned}
\mathbf{P}^{-1}(n_s) = \boldsymbol{\phi}^T(n_s)\boldsymbol{\phi}(n_s) &= \sum_{i=1}^{n_s}\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^T(i) &(5.35)\\
&= \sum_{i=1}^{n_s-1}\boldsymbol{\varphi}(i)\boldsymbol{\varphi}^T(i) + \boldsymbol{\varphi}(n_s)\boldsymbol{\varphi}^T(n_s)\\
&= \mathbf{P}^{-1}(n_s-1) + \boldsymbol{\varphi}(n_s)\boldsymbol{\varphi}^T(n_s)
\end{aligned}$$

where $\mathbf{P}^{-1}(n_s-1)$ can be computed as,

$$\mathbf{P}^{-1}(n_s-1) = \mathbf{P}^{-1}(n_s) - \boldsymbol{\varphi}(n_s)\boldsymbol{\varphi}^T(n_s) \tag{5.36}$$

The final recursive equations are summarized as follows.

$$\hat{\boldsymbol{\eta}}(n_s) = \hat{\boldsymbol{\eta}}(n_s-1) + \mathbf{K}(n_s)[\hat{f}(n_s) - \boldsymbol{\varphi}^T(n_s)\hat{\boldsymbol{\eta}}(n_s-1)] \tag{5.37}$$

where

$$\begin{aligned}
\mathbf{K}(n_s) &= \mathbf{P}(n_s)\boldsymbol{\varphi}(n_s)\\
&= \mathbf{P}(n_s-1)\boldsymbol{\varphi}(n_s)[\mathbf{I} + \boldsymbol{\varphi}^T(n_s)\mathbf{P}(n_s-1)\boldsymbol{\varphi}(n_s)]^{-1} &(5.38)\\
\mathbf{P}(n_s) &= [\mathbf{I} - \mathbf{K}(n_s)\boldsymbol{\varphi}^T(n_s)]\mathbf{P}(n_s-1) &(5.39)
\end{aligned}$$

5.4 Conclusions

In this section, the ESO is adopted to estimate the unknown dynamics. Furthermore, the observability of the ESO for reconstructed extended observer system has been investigated in detail. The special structure of the ESO can help us find the minimum number of sensors to estimate the unknown dynamics. Once the unknown dynamics are estimated, the recursive least squares algorithm is adopted to identify the system parameters.

Chapter 6

ENGINEERING APPLICATION I

6.1 High Speed Train Bogie

To date, high-speed train plays an important role in the transportation and has become the fastest ground-level infrastructure worldwide. Dynamics and control of the high-speed train for the improvement of stability and safety has attracted tremendous attention from scholars and engineers [101–103]. As a crucial subsystem of the high-speed train, the bogie interacts with the rail through the suspension and heavily influences the stability of the system. The bogie has quite a few degrees of freedom, and is usually quite restricted in terms of the placement of sensors and actuators. Furthermore, the interaction forces between the wheel and rail are typically nonlinear and destabilizing when the speed of the train is high enough. As the bogie is an underactuated mechanical system with complex interactions with the environment, it remains quite challenging to develop effective controls for improving the stability of the train at a high speed. For example, the actuators installed on the bogie do not have direct controls over the wheel-rail interactions which are responsible for the hunting instability. The control must take advantages of dynamic coupling of different degrees of freedom in order to increase the train speed while maintaining stability. This chapter presents a control design making use of the balanced truncation reduced order model, which assists us in finding the dominant coupled dynamics of the bogie and thus streamlines control design.

There have been many control studies of the bogie [104, 105], in particular, to improve the hunting stability [2, 106]. The semi-active controls of the suspension have been studied including the H_∞ control [107] and optimal control with magnetorheological (MR) dampers [108]. These semi-active controls also improve the riding comfort. Other active strategies using feedback controls to enhance the hunting stability of the bogie have also been developed [104, 105, 109]. The genetic algorithm NAGA-II is adopted to optimize controls for improving the hunting stability [2] and to find the optimal suspension parameters [110].

Not much attention has been paid on exploration of the nature of being underactuated as well as the high dimensions of the bogie in the literature. To address these issues, this section presents a new control approach by developing an extended balanced truncation model reduction (BTMR) method to increase the critical train speed while keeping the bogie stable. The BTMR has been applied to control design for high dimensional systems with thousands of degrees of freedom [111, 112]. The BTMR method is applied to study the dynamics of a high-speed bogie in [113], where the responses of the reduced order system is compared to those of the full order model. However, the BTMR method has not been adopted for the control design of the bogie in the literature. This section introduces the BTMR method to design control for high dimensional underactuated multi-input-multi-output (MIMO) bogie system. The main contributions of this chapter are summarized below.

1. The BTMR method is introduced to deal with the underactuated property of the high dimensional bogie system with unstable dynamics.
2. The balanced truncation (BT) reduced order model of the system is created that captures the system dynamics accurately, and provides a platform for control design.
3. The BTMR method with non-zero initial conditions is extended to the control design problem of underactuated mechanical systems.

6.1.1 Model of Bogie

The bogie satisfies the equations of motion of Newton's law in reference to the compartment. This formulation implies an assumption that the compartment body moves with a constant speed in a straight path and that its dynamics is ignored in the description of the bogie motion. In particular, we consider the model of the bogie with 8 degrees of freedom as reported in [2].

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}_D\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} = \mathbf{E}\mathbf{u} \tag{6.1}$$

where $\mathbf{z} = [y_{w1}, \varphi_{w1}, y_{w2}, \varphi_{w2}, y_f, \varphi_f, y_{m1}, y_{m2}]^T \in \mathbb{R}^{8\times 1}$ consists of eight linear and rotational displacements of the bogie. $\mathbf{u} \in \mathbb{R}^{2\times 1}$ are two controls as indicated in Figure 6.1. $\mathbf{M} \in R^{8\times 8}$ is the symmetric and positive definite mass matrix. $\mathbf{C}_D \in \mathbb{R}^{8\times 8}$ is the symmetric semi-positive definite damping matrix. $\mathbf{K} \in \mathbb{R}^{8\times 8}$ is the stiffness matrix. $\mathbf{E} \in \mathbb{R}^{8\times 2}$ is the matrix describing the influence of controls on the motion of the bogie. The stiffness matrix as given in [2] contains the rail-wheel interaction forces which destroy the symmetry of the matrix. This is the source of instability of hunting motion. When the system has enough damping when the train travels at relatively low speed, the dynamics is still stable. When the train speed increases beyond a critical value, the damping decreases enough so that the rail-wheel interaction forces destabilize the system and cause hunting motion.

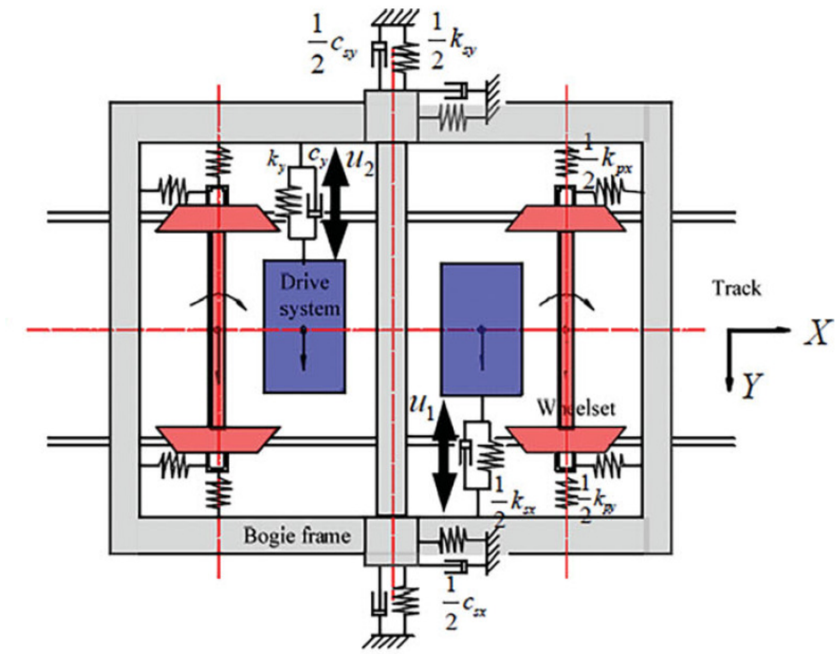


Figure 6.1: Dynamic model of bogie system (Picture from [2]).

Define a state vector as $\mathbf{x} = [\mathbf{z}, \dot{\mathbf{z}}]^T \in \mathbb{R}^{16\times 1}$. We arrive at a LTI system in the state space,

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{6.2}$$

where $\mathbf{y}$ is a vector of measured outputs of the system. The matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are given by

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 0 & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C}_D \end{bmatrix} \in \mathbb{R}^{16\times 16} \\ \mathbf{B} &= \begin{bmatrix} 0 \\ \mathbf{M}^{-1}\mathbf{E} \end{bmatrix} \in \mathbb{R}^{16\times 2},\ \ \mathbf{C} = [\mathbf{I}, \mathbf{0}] \in \mathbb{R}^{4\times 16} \end{aligned} \tag{6.3}$$

All the definitions of displacements, system parameters and matrices of the bogie can be found in [2]. By examining matrices $\mathbf{A}$ and $\mathbf{B}$, it is obvious that the bogie is a relatively high dimensional, strongly coupled underactuated system. In Section 3.1, we introduce the BTMR method to find a reduced order model of the system, which keeps the main features of the original system and is amenable to the control design. We should also note that the condition for existence of the unique solution of Equation (3.4) in Chapter 3 is satisfied by the bogie model as we find out in the numerical simulations.

6.1.2 BT Reduced Order Model of the Bogie

We now apply the BTMR method to the bogie. Recall that some components of the damping matrix $\mathbf{C}_D$ of the bogie decreases as the train speed increases. When the speed is sufficiently high, the wheel-rail interaction terms as asymmetric parts of the stiffness matrix drive the train into hunting instability. Hence, we must apply the extended BTMR method presented earlier to the system with unstable dynamics.

There are many ways to investigate the instability as a function of the train speed. Since we always compute the eigenvalues of the bogie, we can use the relative stability criterion, i.e. the damping ratio of the complex eigenvalues, to find the critical speed of the bogie. The minimum damping ratio of all the eigenvalues is calculated with respect to the train speed. Figure 6.13 shows the damping ratio as a function of the train speed for various cases. The minimum damping ratio is equal to zero when the train speed is said to be critical. For the current bogie system, the open-loop critical train speed is found to be 415 km/h.

In the following, we show two examples of application of the BTMR method to the bogie. The first one is for the stable case when the train speed $v = 360\ km/h$ is below the critical speed. The Hankel singular values, transformation matrix $\mathbf{T}$ and the balanced truncation reduced order model of the bogie can be obtained directly. In all the computations, we have chosen $\mathbf{X}_0$ to be the unit matrix as discussed previously. From Equation (3.26), we know that the norm of the output error is bounded. Figure 6.2 shows the norm of the original system output and reduced model output in open loop. The difference between the two is indeed bounded and the norm of error between the outputs is also shown in Figure 6.2. The Hankel singular values of the bogie are shown in Figure 6.5. Recall that the bogie model has 8 degrees of freedom, i.e. 16 states. Figure 6.5 suggests that the last twelve states with low energy can be truncated. This result also implies that when the bogie is stable, there are only two modes which dominate the system response. This further suggests that only two properly placed controls may be sufficient to control the dominant dynamics of the bogie.

The physical degrees of freedom contributing to the two stable dominant modes can be identified as follows.

Recall the transformation of the BT method in Equation (3.20). We have

$$\mathbf{A} = \mathbf{T}\mathbf{A}_{BT}\mathbf{T}^{-1} \tag{6.4}$$

Apply the modal decomposition to the matrix $\mathbf{A}_{BT}$. We have

$$\mathbf{A}_{BT} = \mathbf{M}^{-1}\mathbf{D}\mathbf{M} \tag{6.5}$$

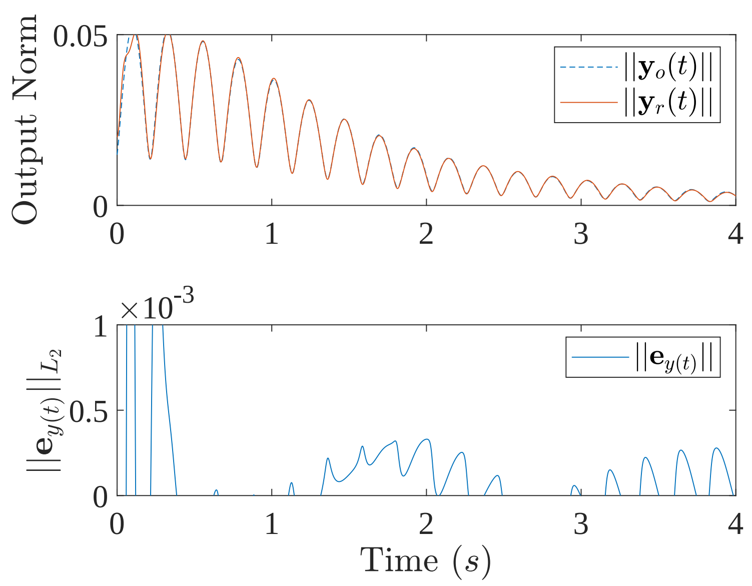


Figure 6.2: Top: Comparison of the L_2 norm of the original system and reduced order model output in stable case. The two outputs are almost indistinguishable. Bottom: The L_2 norm of the output error.

where $\mathbf{M}$ is the modal matrix of $\mathbf{A}_{BT}$ and $\mathbf{D}$ is a diagonal matrix consisting of eigenvalues of $\mathbf{A}_{BT}$. Hence, the matrix $\mathbf{A}$ reads

$$\mathbf{A} = \mathbf{T}\mathbf{M}^{-1}\mathbf{D}\mathbf{M}\mathbf{T}^{-1} = \mathbf{H}^{-1}\mathbf{D}\mathbf{H} \tag{6.6}$$

where $\mathbf{H} = \mathbf{M}\mathbf{T}^{-1}$. The columns of the matrix $\mathbf{H}$ are the modal eigenvectors of $\mathbf{A}$, while the diagonal matrix $\mathbf{D}$ consists of the eigenvalues of $\mathbf{A}$. By examining the matrix $\mathbf{D}$, we can identify the two stable dominant modes in the transformed system. The modal participation of the physical degrees of freedom in the two dominant modes is shown in Figure 6.3. It is clear that the two dominant modes consist of special linear combinations of the physical degrees of freedom.

The second example considers the unstable bogie when the train speed is beyond the critical speed 415 km/h. We select $v = 432$ km/h. We must apply the extended BTMR method discussed earlier. Figure 6.4 shows the comparison between the original system output and the reduced order system output. The error bound in Figure 6.4 shows that the reduced order system is a good approximation of the original system in terms of the output for unstable case. This allows us to further investigate the control design probability by making use of the reduced order system in later section. Figure 6.6 shows the Hankel singular values. The two stars represent the unstable modes of the system with infinite values, which are actually out of the bounds of the plot. The circles represent the stable modes of the bogie

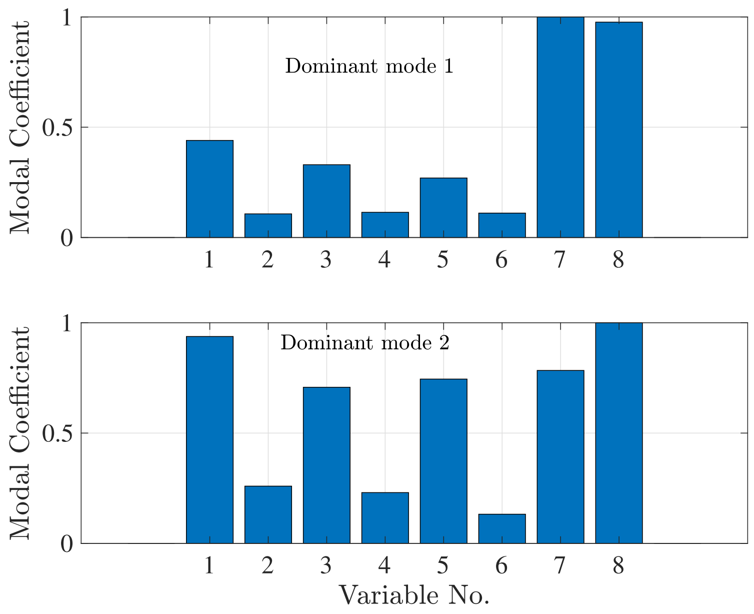


Figure 6.3: Modal participation of the physical degrees of freedom to the two dominant stable modes in the transformed system. The variable number refers to the position of various displacements in the vector $\mathbf{z} = [y_{w1}, \varphi_{w1}, y_{w2}, \varphi_{w2}, y_f, \varphi_f, y_{m1}, y_{m2}]^T$.

with finite singular values.

If we choose the same truncation criterion by considering the Hankel singular values, we would keep 6 states, i.e. two unstable states and the first four stable states. The reduced order model is thus of order six. To be able to fully control these states, we would need three properly designed controls. On the other hand, if we continue to keep only four states, i.e. two unstable and two stable states, the reduced order model would be of order four. Two controls could fully regulate the motion of the reduced order model dynamics.

Finally, we point out that the BT reduced order model retains the unstable states and has the same stability properties as the original system.

6.1.3 Simulations of Bogie Control

In this section, we shall simulate the control with the Luenberger observer designed in Section 3.3.2. We consider both the stable and unstable cases of the bogie. To better illustrate the performance of the control, a numerical criterion is introduced. Assume that all the states converge to zero under the control. The numerical criterion is defined as the settling time t_s such that for $t \geq t_s$, $||\mathbf{x}(t)||_{L_2} \leq 0.01$.

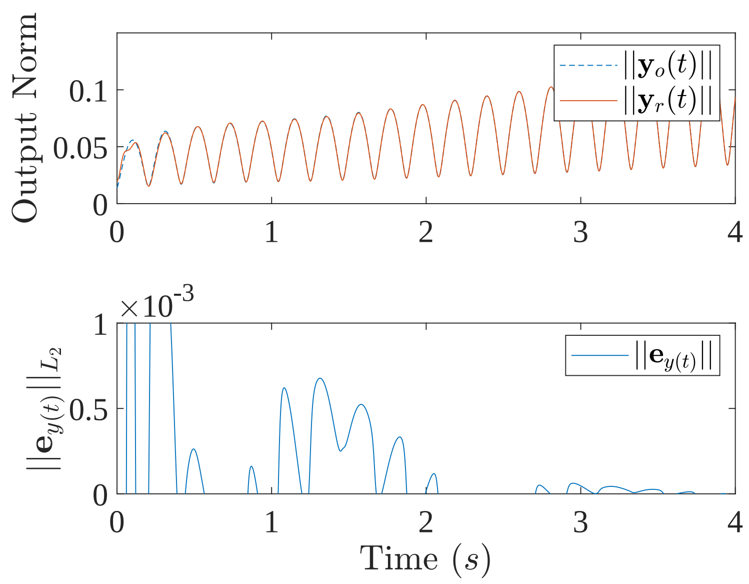


Figure 6.4: Top: Comparison of the L_2 norm of the original system and reduced order model output in unstable case. The two outputs are almost indistinguishable. Bottom: The L_2 norm of the output error.

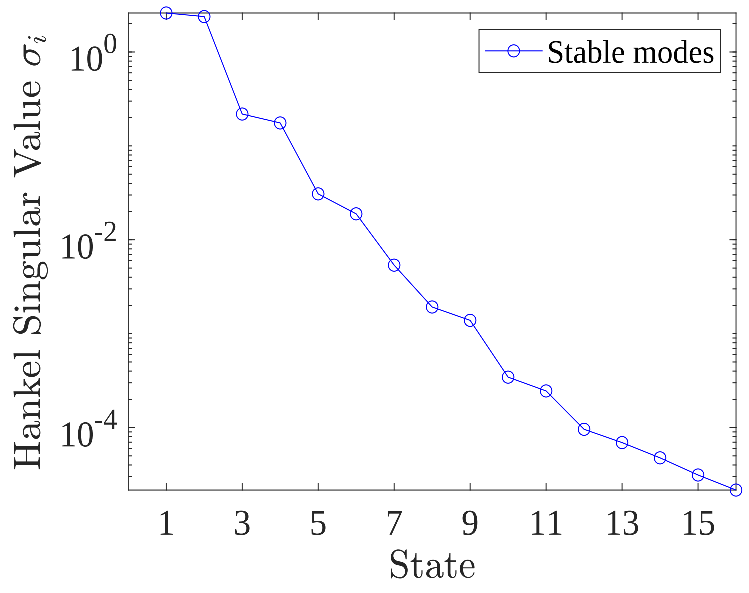


Figure 6.5: Hankel singular values of the stable bogie when the train speed $v = 360$ km/h is below the critical speed.

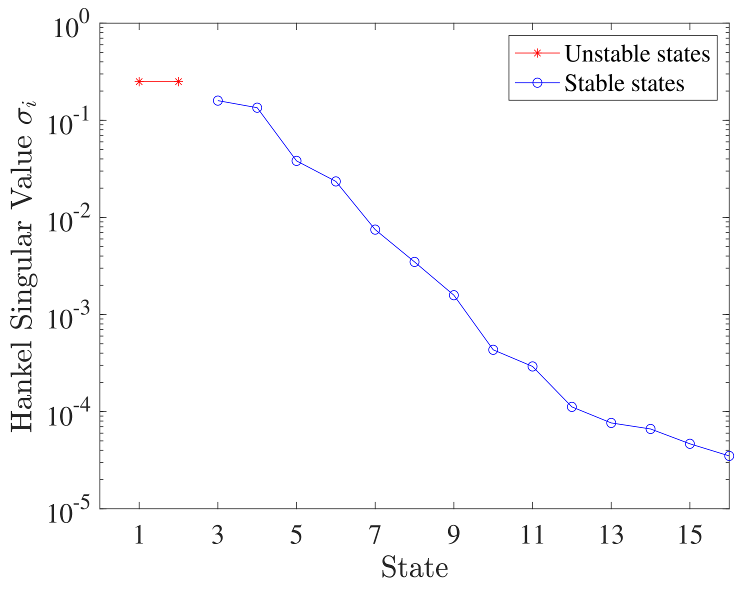


Figure 6.6: Hankel singular values of the unstable bogie when the train speed $v = 432\ km/h$ is beyond the critical speed. The two stars represent the unstable modes of the system with infinite values.

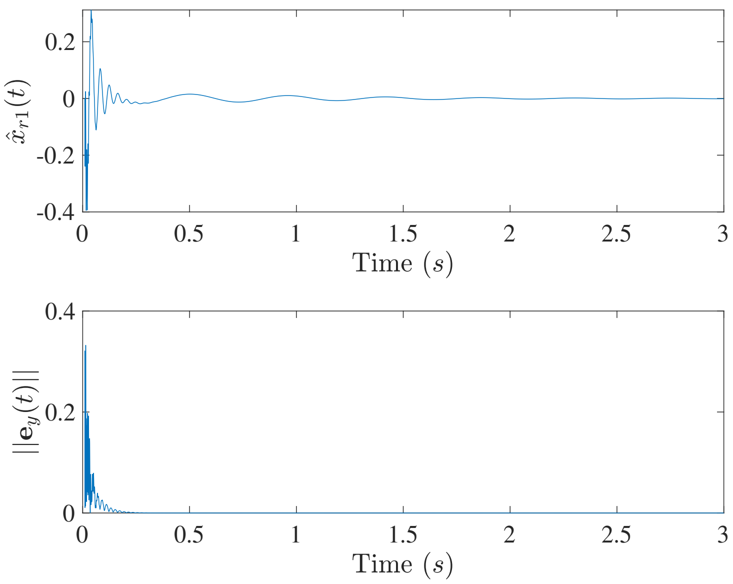


Figure 6.7: Example of estimated states and estimation errors when the train speed is $v = 360km/h$. Top: The estimated state $\hat{x}_{r1}(t)$. Bottom: The norm of the estimation errors $\mathbf{e}_y(t)$.

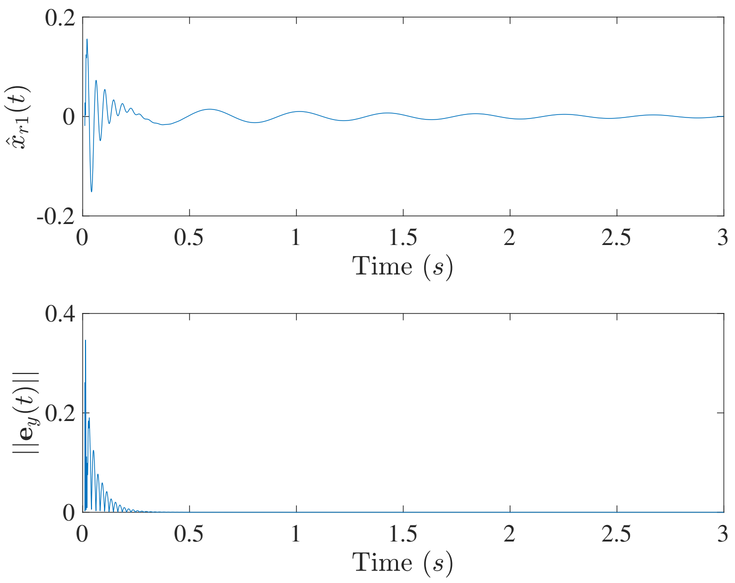


Figure 6.8: Example of estimated states and estimation errors when the train speed is $v = 432km/h$. That is, the open-loop is unstable. Top: The estimated state $\hat{x}_{r1}(t)$. Bottom: The norm of the estimation errors $\mathbf{e}_y(t)$.

6.1.3.1 Stable Case

Let the train velocity be 360 km/h less than the critical speed 415 km/h. The initial conditions are $x_i(0) = 0.01$ for $1 \leq i \leq 16$. The matrices for the LQR control are selected as,

$$\mathbf{Q} = 100000\mathbf{I}, \mathbf{R} = \begin{bmatrix} 10^{-6} & 0 \\ 0 & 10^{-6} \end{bmatrix} \tag{6.7}$$

where $\mathbf{I} \in \mathbb{R}^{4\times4}$ is the identity matrix. The observer gain $\mathbf{L}$ is designed to make the state estimation converge fast than the closed-loop pole of the control. In the examples reported later, we have chosen the closed-loop poles of the observer at $-245.5 \pm i1234.6$. Note that this is a double pole, and 10 to 100 times faster than the closed-loop poles of the control.

Figure 6.9 shows the time history of the two controls and the first state $x_1(t)$ of both the open-loop and closed-loop system. Other states have similar behavior and are not shown for the sake of clarity.

The control drives the states to zero faster than the naturally damped free response. The norms of the state vector of the closed-loop and open-loop system are shown in Figure 6.10. In this case, the control leads to a settling time $t_s = 4.10$ seconds as compared to $t_s = 6.14$ seconds of the free response.

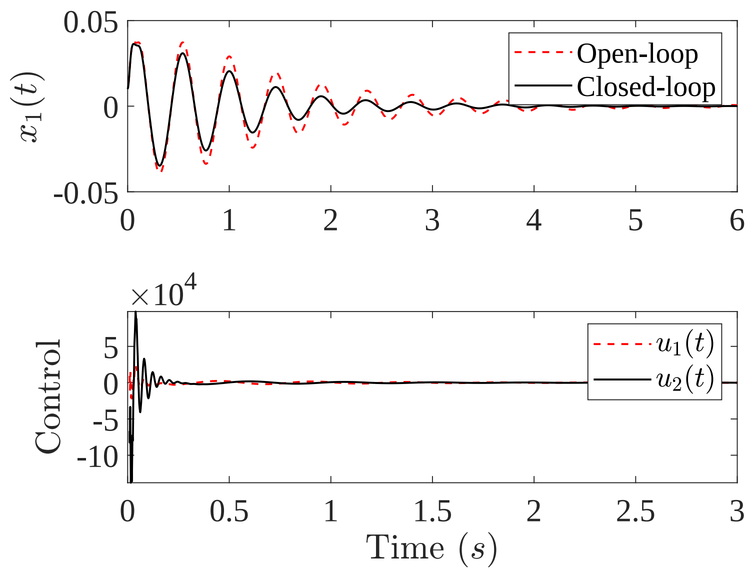


Figure 6.9: Controls and state $x_1(t)$ of the stable bogie.

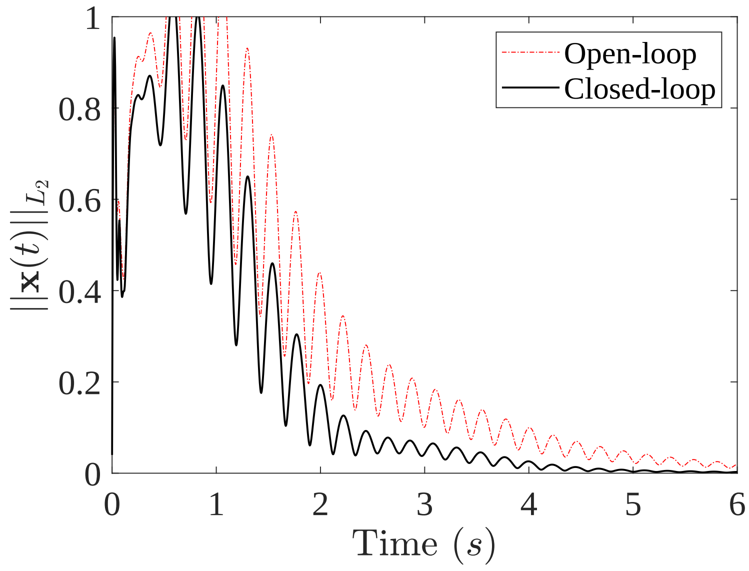


Figure 6.10: Comparison of the norms of the open-loop and closed-loop state $\mathbf{x}(t)$ for the stable bogie.

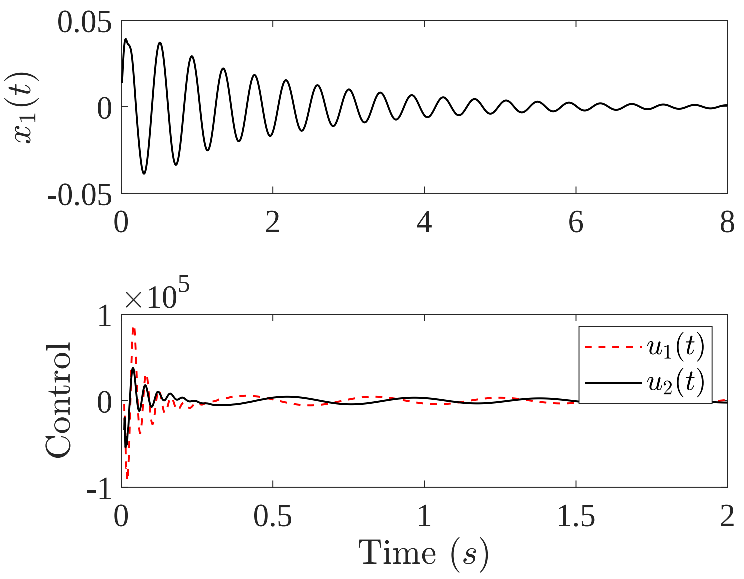


Figure 6.11: Controls and state $x_1(t)$ of the unstable bogie.

6.1.3.2 Unstable Case

Let the train speed be $v = 432\ km/h$. For the unstable bogie, we keep the same initial conditions, matrices Q and R as for the stable bogie. Figure 6.11 shows the controls and state $x_1(t)$. Results show that the BTMR method works well to stabilize the unstable bogie system and to further improve the hunting stability of the train at a speed higher than the open-loop critical speed 415 km/h.

In simulations, it takes about 8.78 seconds for the controlled bogie system to settle in the range $||\mathbf{x}||_{L_2} \leq 0.01$. Recall that the response of the open-loop system grows unbounded as time increases. The L_2 norm of the states of the closed-loop system is shown in Figure 6.12. The figure indicates that all the states of the closed-loop system are stabilized.

6.1.3.3 Robustness and Stability Improvement

Results in Section 6.1.3.1 and Section 6.1.3.2 prove that the LQR control is effective in improving the hunting stability of the bogie at high speed. It should be pointed out that the train speed can be viewed as a design parameter when the LQR control is considered. Since the train speed changes during operation, and there are disturbances and uncertainties, it is necessary to examine the robustness of the control. One question we like to address is: If the control is designed for a given train speed, is the closed-loop system still stable for other speeds, particularly for higher speed than the design targeted speed?

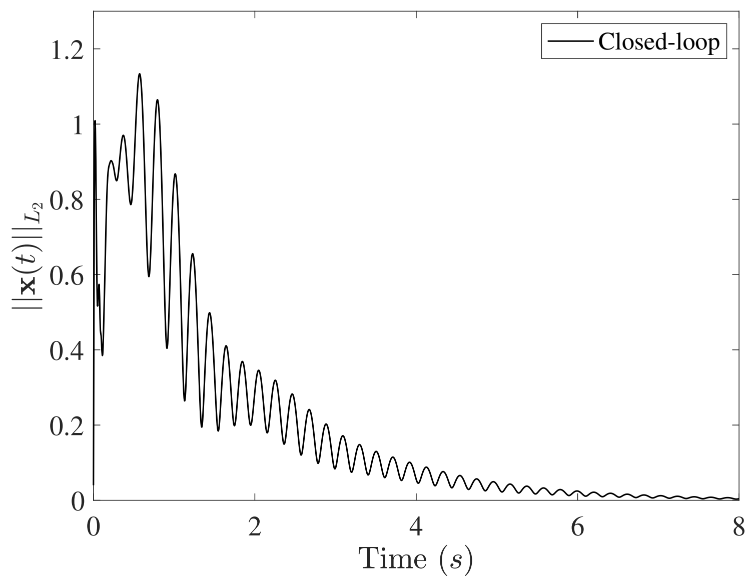


Figure 6.12: Norm of the closed-loop state $\mathbf{x}(t)$ for unstable case. The norm of the open-loop state grows unbounded.

With the control law in Equation (3.35), the closed-loop system can be written as,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}(t) - \mathbf{B}\mathbf{K}_{opt}\mathbf{x}_r(t) \tag{6.8}$$

Recall the total transformation $\mathbf{x} = \mathbf{T}\mathbf{x}_{BT}$ where

$$\mathbf{x}_{BT} = \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_t \end{bmatrix} = \mathbf{T}^{-1}\mathbf{x}. \tag{6.9}$$

It is noted again that the reduced order state vector $\mathbf{x}_r$ in the above equation may contain unstable states if the transformation $\mathbf{T}$ is designed for the bogie traveling at speed above 415 km/h. Let us partition the inverse transformation $\mathbf{T}^{-1}$ as

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{T}_r^{-1} \\ \mathbf{T}_t^{-1} \end{bmatrix}. \tag{6.10}$$

where $\mathbf{T}_r^{-1} \in \mathbb{R}^{r \times n}$. Hence, we have

$$\mathbf{x}_r = \mathbf{T}_r^{-1}\mathbf{x}. \tag{6.11}$$

The closed-loop system matrix $\mathbf{A}_{cl}$ becomes,

$$\mathbf{A}_{cl} = \mathbf{A} - \mathbf{B}\mathbf{K}_{opt}\mathbf{T}_r^{-1} \tag{6.12}$$

The variation of the closed-loop minimum damping ratio ζ_{min} with the train speed is shown in Figure 6.13. It is seen from the figure that the control designed for a specific train speed keeps the bogie stable for a wide range of train speed including the speed higher than the targeted design speed and even higher than the critical speed of the open-loop system. For example, the control designed at speed 432 km/h stabilizes the system up to the speed 490 km/h. This is an excellent control robustness. Hence, the proposed control has potential to increase the train speed without hunting instability.

Figure 6.13 also shows that ζ_{min} achieves a maximum at some speed, which represents the highest stability margin of the given control. An interesting question to ask is: Can we design the control such that the maximum of ζ_{min} occurs at a desired train speed? With a proper choice of $\mathbf{Q}$ and $\mathbf{R}$ matrices in the LQR formulation such as,

$$\mathbf{Q} = 100000\mathbf{I}, \mathbf{R} = \begin{bmatrix} 10^{-7} & 0 \\ 0 & 10^{-7} \end{bmatrix} \tag{6.13}$$

we can indeed place the maximum of ζ_{min} at 400 km/h, for example. The dashed curve in Figure 6.13 shows the result of the example. An optimization procedure is yet to be developed to do this kind of control design systematically.

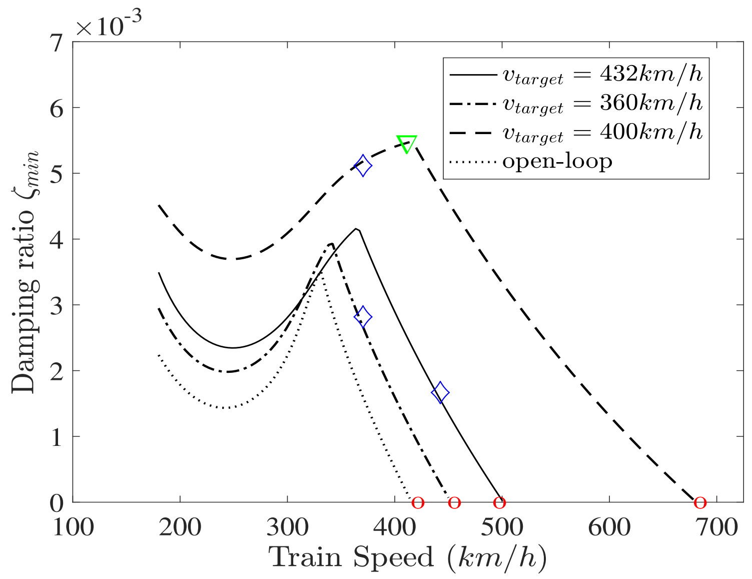


Figure 6.13: The minimum damping ratio ζ of all the eigenvalues of the matrix $\mathbf{A}_{cl}$. The diamonds represent different targeted train speeds for the control design. The circles denote the open-loop and closed-loop critical speeds of the bogie. The higher dashed line with ∇ symbol represents the case when the control is designed for the speed $v_{target} = 400km/h$ with the choice of $\mathbf{Q}$ and $\mathbf{R}$ in Equation (6.13). The dotted line indicates the open-loop critical speed $v = 415\ km/h$. The positive and negative symbols of ζ_{min} were switched to better illustrate the figure.

6.1.4 Conclusions

We have presented a balanced truncation reduced order model for the bogie of high-speed train, which is severely underactuated and difficult to control. The reduced order model allows us to focus on the dominant dynamics of the system and to develop effective controls to improve the hunting stability and increase the critical speed. Although various methods for model reduction have been well studied in the literature, the application of these methods to such a complex and multi-degree-of-freedom underactuated dynamic system is rare. The reduced order model opens the opportunities to choose various controls for the bogie. As an example, the LQR optimal control is designed for the reduced order model of the bogie together with a Leunberger observer. The control is then implemented on the full model of the bogie. Extensive numerical simulations have been presented and suggest that the control designed with the help of the reduced order model can substantially improve the hunting stability over a wide range of train speeds. This indicates a strong robustness of the control with respect to the train speed, as measured by the stability robustness measure introduced in this section. We have also examined the effect of the train speed, which is pre-selected in the control design. An optimization problem can be formulated such that the closed-loop system has the largest stability margin at a preferred speed. This will be the topic of a subsequent study.

6.2 Wheel-rail Interactions Estimation

Wheel-rail contact forces are usually nonlinear, motion dependent, time varying, and destabilizing. The wear, irregularity, and fatigue of the wheel and rail are primarily due to the wheel-rail contact forces. As interest in high-speed trains continue to increase, stability and the safety of operations have received much attention from the research community. It has become apparent that in order to design effective controls to improve the performance of the train in motion, and to monitor the health of the wheel and rail, we need a good knowledge of the wheel-rail contact forces. However, these contact forces cannot be directly measured when the train travels at high speed. Effective estimation of the wheel-rail contact dynamics from the train motion signals becomes a highly appealing approach. This section presents a method to estimate the wheel-rail contact forces and the corresponding parameters of a wheel-rail contact force model by making use of the motion measurements of a bogie.

There have been many studies of estimation of the wheel-rail contact forces in the literature. Various methods for estimation have been considered, including inverse modeling, a filter based system identification method, and modern machine learning algorithm with neural networks. The inverse identification technique was first proposed and applied to the wheel-rail contact force estimation in [114]. Later, a low-cost and constrained inverse wagon model was developed to estimate the wheel-rail contact forces [115]. The gray box inverse wagon model estimation method [116] was further investigated to overcome the limitations of the white box approach in [115]. The indirect method is another form of inverse modeling. An indirect model-based estimation method was proposed to investigate the wheel-rail contact force based on the derailment criteria [117]. The lateral forces on two different sides of the wheel were estimated through the improved indirect method [118]. The inverse methods are highly dependent upon the system model and are not effective with estimation of the time-varying dynamics.

The filter-based estimation of the wheel-rail interface force was first proposed in [119] using the extended Kalman filter, which was further developed in [120] as a nonlinear estimator and applied in [121] to estimate the lateral track irregularities. The advantage of the extended Kalman filter lies in that the parameters of the vehicle system can be estimated separately, with the help of a linearized model of the system. Kalman filter is a model-based estimation algorithm which requires the full system state to be measured [122]. Kalman filter is still a popular choice for researchers to estimate the wheel-rail contact forces [123, 124]. The Kalman filter, particle filter, and linear extended state observer are three state of the art real-time estimation algorithms [125]. Numerical simulations prove that these three algorithms all have excellent performance. The vast applications of Kalman filter mentioned above indicate that the model-based estimation algorithm is practical and has the potential to estimate wheel-rail dynamics in real time.

Neural networks from the machine learning area is another technique that has been used to estimate the wheel-rail contact forces and track irregularity [126] and to predict the safety of railway vehicles [127]. Neural networks can approximate the nonlinear complex functional relationship between the vehicle response and the wheel-rail contact forces without the need of a detailed model of the system.

The wheel-rail contact forces are inherently nonlinear, and change with the operating environment including temperature, cross winds, rain, *etc.* These make it a challenge to model and estimate the contact forces. Moreover, the wheel-rail contact forces determine the hunting stability of the bogie system and play an important role in the maintenance and fault detection of the high-speed train. Inverse modeling, the filtered-based estimation method, and machine learning algorithm have been extensively applied to estimate the wheel-rail contact force. However, these algorithms have quite limitations on the dynamic model, matrices calculation and system signals. These deficiencies of the existing algorithms motivate us to develop an algorithm to estimate the slow time-varying wheel-rail contact forces by making use of limited system signals without relying on the unknown dynamic model. Furthermore, once the wheel-rail contact forces are estimated, how to make full use of the estimated contact force for maintenance and fault detection of the high-speed train will be another crucial question. In this section, the wheel-rail contact forces combined with the time-varying gravitational stiffness effect are viewed as unknown dynamics. An extended state observer (ESO) is implemented to estimate the unknown dynamics due to the wheel-rail interaction. Moreover, we would like to build a relationship between the motion signals and wear prediction. This has not been covered in the literature and will be a future study topic.

In this section, we adopt the linear ESO discussed in Section 5 to estimate the wheel-rail interactions from motion measurements for a high-speed train bogie. We also investigate the observability of the linear ESO when applied to the bogie system. The main contributions of the section are summarized below.

1. The wheel-rail interactions are first treated as unknown dynamics, which are estimated with an ESO from motion signals. The proposed ESO is proven to be observable and is shown to be able to capture the time-varying wheel-rail interactions with the motion measurements using a limited number of sensors.

2. A regression method is proposed to identify two key parameters of a wheel-rail contact force model: the lateral creep coefficient f_η and gravitational stiffness k_{gy}. The regression model is then used to compute the wheel-rail contact forces. This result lays a foundation for further study of wheel creep damage and fatigue in the future.

6.2.1 Extended State Observer for Bogie System

Consider the eight degree-of-freedom mathematical model of the bogie in [2]

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}_D\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} = \mathbf{K}_{us}\mathbf{z} + \mathbf{E}\mathbf{u} \tag{6.14}$$

where $\mathbf{z} = [y_{w1}, \varphi_{w1}, y_{w2}, \varphi_{w2}, y_f, \varphi_f, y_{m1}, y_{m2}]^T \in \mathbb{R}^{8\times1}$ consists of eight linear and rotational displacements of the bogie. $\mathbf{u} \in \mathbb{R}^{2\times1}$ are two controls. $\mathbf{M} \in R^{8\times8}$ is the symmetric and positive definite mass matrix. $\mathbf{C}_D \in \mathbb{R}^{8\times8}$ is the symmetric semi-positive definite damping matrix. $\mathbf{K} \in \mathbb{R}^{8\times8}$ is the symmetric semi-positive stiffness matrix. $\mathbf{E} \in \mathbb{R}^{8\times2}$ is the matrix describing the influence of controls on the motion of the bogie.

The matrix $\mathbf{K}_{us}$ is defined as

$$\mathbf{K}_{us} = \begin{bmatrix} \mathbf{K}_\lambda & \mathbf{0}_{4\times4} \\ \mathbf{0}_{4\times4} & \mathbf{0}_{4\times4} \end{bmatrix} \in \mathbb{R}^{8\times8} \tag{6.15}$$

where

$$\mathbf{K}_\lambda = \begin{bmatrix} k_{gy} & -2f_\eta & 0 & 0 \\ \frac{2\lambda_e l_0 f_\xi}{r_0} & 0 & 0 & 0 \\ 0 & 0 & k_{gy} & -2f_\eta \\ 0 & 0 & \frac{2\lambda_e l_0 f_\xi}{r_0} & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{6.16}$$

$\mathbf{K}_{us}\mathbf{z}$ describes the wheel-rail interaction forces. The matrix $\mathbf{K}_{us} \in \mathbb{R}^{8\times8}$ is not symmetrical. It is the source of instability of hunting motion. An objective of this work is to develop an extended state observer to estimate the wheel-rail interactions. All the matrices and their numerical values are available in Section 9.1.

We first convert the equation of motion to the state space form as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}_s\mathbf{x}(t) + \mathbf{B}_u\mathbf{u}(t) + \mathbf{B}_f\mathbf{f}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{6.17}$$

Table 6.1: Definition of the parameters in the matrix $\mathbf{K}_\lambda$.

Symbol	Description
k_{gy}	The gravitational stiffness
λ_e	Wheel-rail contact conicity
$2l_0$	Distance of the contact spot
f_ξ	The longitudinal creep coefficient
f_η	The lateral creep coefficient
r_0	The wheel rolling radius

where $\mathbf{x}(t) = [\mathbf{z}, \dot{\mathbf{z}}]^T \in \mathbb{R}^{16\times1}$ is the state vector, and all other matrices are given by

$$\begin{aligned}
\mathbf{A}_s &= \begin{bmatrix} \mathbf{0}_{8\times8} & \mathbf{I}_8 \\ -\mathbf{M}^{-1}\mathbf{K}_s & -\mathbf{M}^{-1}\mathbf{C}_D \end{bmatrix} \in \mathbb{R}^{16\times16}, \\
\mathbf{B}_u &= \begin{bmatrix} \mathbf{0}_{8\times2} \\ \mathbf{M}^{-1}\mathbf{E} \end{bmatrix} \in \mathbb{R}^{16\times2}, \\
\mathbf{B}_f &= \begin{bmatrix} \mathbf{0}_{8\times4} \\ \mathbf{I}_4 \\ \mathbf{0}_{4\times4} \end{bmatrix} \in \mathbb{R}^{16\times4}, \\
\mathbf{M}^{-1}\mathbf{K}_{us}\mathbf{z}(t) &= \begin{bmatrix} \mathbf{f}(t) \\ \mathbf{0}_{4\times1} \end{bmatrix} \in \mathbb{R}^{8\times1}, \quad \mathbf{f}(t) \in \mathbb{R}^{4\times1}, \quad \mathbf{C} \in \mathbb{R}^{m\times16},
\end{aligned} \tag{6.18}$$

We denote $\mathbf{I}_n$ as the $n \times n$ identity matrix. $\mathbf{C}$ is the output matrix. The dimension m of $\mathbf{C}$ is equal to the number of the outputs. The choice of the outputs determines the observability of the system and is discussed in Section 5.2.

6.2.2 Model for the Wheel-rail Interaction

The parameters in the matrix $\mathbf{K}_\lambda$ in Equation (6.16) are defined in Table 6.1. These parameters describe the wheel-rail geometry and material creep behavior. The wheel-rail contact equivalent conicity λ_e in particular is a result of the wheel-rail interaction dynamics. In the linear theory, the wheel-rail contact equivalent conicity λ_e is taken to be a constant. In the nonlinear theory, λ_e is usually taken as a time-varying nonlinear function of the lateral displacement of the wheel set $y_w(t)$ [128]. The gravitational stiffness k_{gy} is a function of λ_e and the weight given by,

$$k_{gy} = \frac{\lambda_e W}{10} \tag{6.19}$$

where W is the weight of the bogie.

It is apparent that the parameters in the matrix $\mathbf{K}_\lambda$ are all dynamic and may change with time. Hence, the unknown dynamics $\mathbf{f}(t)$ in Equation (6.17) cannot be determined a priori. The unknown dynamics $\mathbf{f}(t)$ must be estimated in real-time in order to design effective controls to stabilize the bogie at high speeds. To this end, we propose an extended state estimation (ESO) algorithm to estimate $\mathbf{f}(t)$.

6.2.3 Simulations of Bogie Dynamics Estimation

In this section, the ESO for the bogie is simulated to estimate the unknown dynamics in the system. The bogie is controlled using feedback controller.

6.2.3.1 Observability of ESO for the Bogie

We have observed difficulty that the state matrix $\mathbf{A}_e$ is ill-conditioned as mentioned in chapter 5.2 with the state matrix of the bogie. The conditions of Lemma 5.2.2 must be checked numerically in order to determine the observability of the ESO for general dynamic systems. Introduce the output matrix $\mathbf{C}$ of bogie system as,

$$\mathbf{C} = [\mathbf{I}_4, \mathbf{0}_{4\times 12}] \tag{6.20}$$

For the bogie problem, we can numerically verify that indeed the ESO for the bogie satisfies the condition 1 of Lemma 5.2.2. The observability matrix $\mathcal{O}_\lambda$ is full rank when the row number m of matrix $\mathbf{C}$ satisfies the condition $m \geq p = 4$, and the second condition of Theorem 5.2.1 $\mathbf{C}_e a \neq 0$ is also numerically checked satisfied for the bogie system. Hence, the conditions in Theorem 5.2.1 are all satisfied to conclude that the ESO for the bogie is observable.

The numerical results of the PBH test for observability are summarized in Table 6.2. It is seen from the table that the computed rank of the observability matrix with ill-conditioned system matrices is not reliable. For example, theoretically if the number of the output m is equal to the degrees of freedom 8 of the bogie, the system must be observable. However, the rank of the observability matrix indicates that the system is not observable.

The PBH test avoids the numerical difficulties in dealing with ill-conditioned matrices. The PDH test results indicate that the extended state system for the bogie can be observable with limited sensors. The ESO for the bogie can indeed estimate the unknown dynamics. This will be demonstrated later.

Table 6.2: Numerical results of two observability criterions for the extended state of the bogie.

No. of output m	rank($\mathcal{O}$)	rank($\mathcal{O}_\lambda$)
3	7	19
4	9	20
6	9	20
8	9	20

6.2.3.2 Estimation of Unknown Dynamics

Next, we report the results of estimation of the unknown wheel-rail contact dynamics. We select $m = 6$ as an example. The train speed is $v = 360km/h$. The observer gain $\mathbf{L}$ is chosen such that the observer poles are 8 to 10 times faster than the poles of the closed-loop control system. The accuracy of the ESO estimation is examined in terms of the norm of the output error defined as,

$$||\mathbf{e}_y(t)|| = ||\hat{\mathbf{y}}(t) - \mathbf{y}(t)|| \tag{6.21}$$

where $|| \cdot ||$ denotes the L_2 norm.

The top sub-figure of Figure 6.14 compares the estimated and true value of the first component of the output vector $\hat{y}_1(t)$ and $y_1(t)$. Note that $y_1(t) = y_{w1}(t)$. The estimated output $\hat{y}_1(t)$ tracks the true value closely. The error $\mathbf{e}_y(t)$ of the entire output vector is shown in the bottom sub-figure of Figure 6.14. The error is of order 10^{-5}.

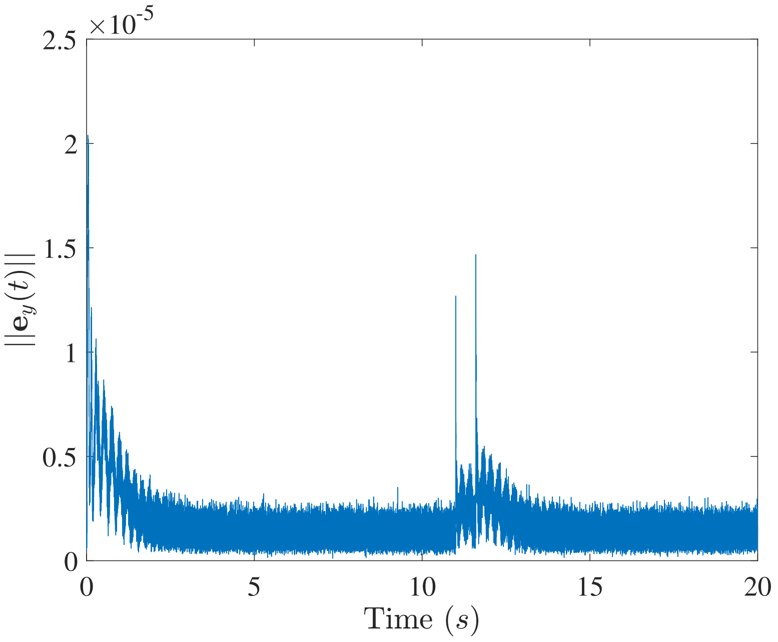


Figure 6.14: Top: Comparison of the estimated output $\hat{y}_1(t)$ from the ESO with the reference system output $y_1(t)$ at the train speed $v = 360$ km/h. Bottom: The norm of the output error vector $\mathbf{e}_y(t)$.

The estimated wheel-rail contact dynamics are shown in Figure 6.15. For each component of wheel-rail contact dynamics $\mathbf{f}(t)$, the estimated contact dynamics $\hat{\mathbf{f}}(t)$ closely follow the reference $\mathbf{f}(t)$. These and other simulation results indicate that the ESO can indeed estimate the outputs and states with a high accuracy, and the extended state $\mathbf{f}(t)$, i.e. the wheel-rail contact dynamics with small bounded errors.

We also investigate the robustness of the ESO to disturbances. At $t = 11s$, a square-impulse disturbance $d(t)$ is introduced as,

$$d(t) = 5 \times 10^{-3}(H(t-11) - H(t-11.6)) \tag{6.22}$$

where $H(t)$ is the Heaviside step function. The results in Figure 6.15 show that the observer tracks the output and wheel-rail contact dynamics quickly and is hence robust to disturbance. To further test the performance, the ESO is applied to the bogie at different train speeds: $v = 100\ km/h$ and $v = 200\ km/h$. The results are shown in Figures 6.16 and 6.17.

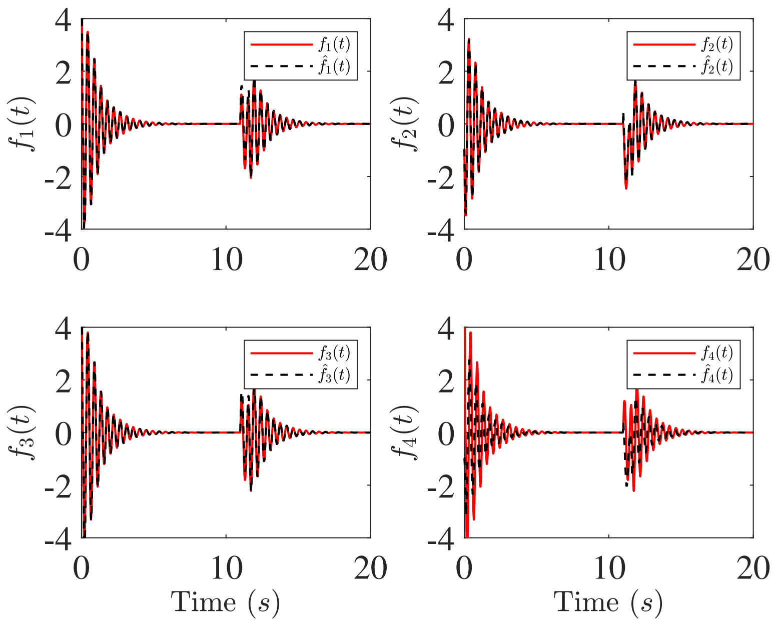


Figure 6.15: Comparison of the estimated wheel-rail contact dynamics $\hat{\mathbf{f}}(t)$ from the ESO with the reference $\mathbf{f}(t)$ at train speed $v = 360\ km/h$.

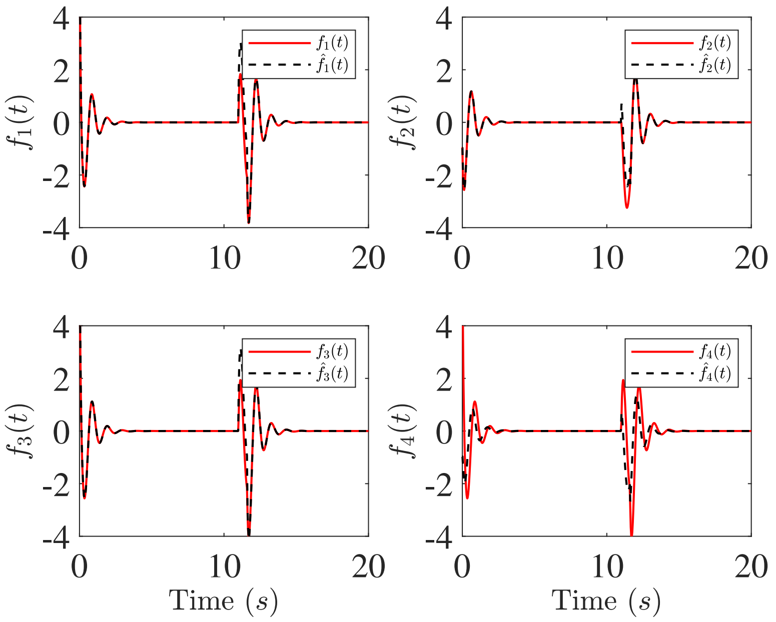


Figure 6.16: Comparison of the estimated wheel-rail contact dynamics $\hat{\mathbf{f}}(t)$ from the ESO with the reference $\mathbf{f}(t)$ at train speed $v = 100\ km/h$.

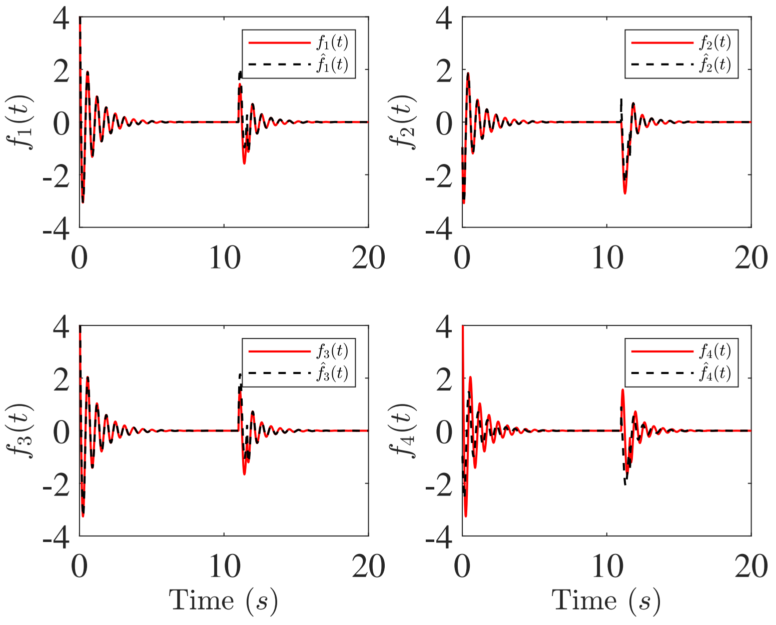


Figure 6.17: Comparison of the estimated wheel-rail contact dynamics $\hat{\mathbf{f}}(t)$ from the ESO with the reference $\mathbf{f}(t)$ at train speed $v = 200\ km/h$.

6.2.4 Conclusions

In this section, we have developed a method to estimate the wheel-rail contact forces of the high-speed train bogie from its motion measurements. The extended state observer is used to estimate the wheel-rail contact dynamics, from which the components of the contact forces can be computed. We have studied the condition for observability of the extended state observer as a function of number of motion sensors. The observed wheel-rail contact forces together with the corresponding motion measurements are then used to estimate parameters of the wheel-rail contact force model. These results can help engineers to monitor the health of wheel and rail from motion signals. We should note that the knowledge of wheel-rail contact forces provides a foundation for wheel damage prediction. This will be the topic of a separate study. Extensive numerical simulations have been done. The results indicate that the extended state observer delivers excellent estimation performance for the bogie with a limited number of motion measurements. Finally, we should point out that the study reported in this section is built on the knowledge of the simplified mathematical model of the bogie. The algorithm is proved to be efficient. The methodology developed in this work is applicable to a train compartment when a complex nominal model of the system is available.

6.3 Wheel Wear Prediction

Maintenance of high-speed train includes wear inspection of the wheel tread. The inspection is usually done over 30 thousand kilo-meters or a fixed time interval [129]. The maintenance schedule of wheel wear induces labor and equipment cost. An efficient and reliable maintenance schedule of the wheel relies on the accurate estimation of the wear. This section proposes a wheel wear estimation method. The method makes use of the estimated external disturbances of a control observer in real-time from motion measurements, and updates a solid mechanics model of wheel wear with estimated model parameters. The resulting model of wheel wear is much more reliable and is used as a surrogate to simulate extensive wheel wear data. The simulated wear data are then used to develop a neural networks model that can predict the wheel wear directly from motion measurements of the train.

Estimation of wheel wear requires a knowledge of dynamic interaction forces between the wheel and rail. Since the direct measurement of the interaction forces in real-time is impossible, methods to estimate them are highly desirable to have. In an early control study of stabilization of the train bogie [33], we developed an extended state observer (ESO) to estimate the state variables of the bogie dynamics as well as the unknown external disturbances which include the wheel-rail interaction forces. The ESO was first proposed in [89] and is one of the core concepts in the active disturbance rejection control (ADRC) [90,91]. This section explores the application of the estimated interaction forces to the wheel wear problem.

Given the interaction forces, a wear model is needed to estimate the creep and wear of the material. There have been many studies in the literature on contact mechanics, creep and wear of solids. The contact theories due to Hertz [130] and Kalker [131] are the foundation for most theoretical studies. A brief survey on the wheel-rail contact mechanics can be found in [132]. Archard's wear model [133] and energy dissipation method [134] are commonly adopted for wheel wear prediction. The energy dissipation method is used to study the two-dimensional wheel-rail contact problem with friction [135]. Archard's wear model is used to predict the wear of commuter rail network in Stockholm. The predictions are compared with the measured wheel profiles [136]. Archard's wear model is also applied to the wheel wear prediction of high speed trains [137,138]. In this section, we shall adopt Archard's wear model as appeared in [136–138].

The wheel-rail interaction forces are nonlinear functions of the train motion and the geometry of the rail. These forces are inherently time-varying and temperature dependent, and are heavily influenced by environmental conditions such as cross wind, rain, snow, and ice. Furthermore, the parameters in Archard's wear model are usually determined under certain experimental conditions. The nominal values of these parameters from the literature or handbook are likely different from the real ones of the train operating in complex environments. For this reason, we propose an algorithm using motion data to update those parameter in order to improve the accuracy of wheel wear prediction.

Since the wheel-rail interaction forces cause the wear and are functions of the train motion and rail geometry, the wheel wear must also be functions of the train motion and rail geometry. However, such functions are implicit and highly nonlinear. In the second part of this section, we apply Archard's wear model together with the estimated wheel-rail interaction forces to simulate wheel wear damages. We then use simulated motions and damages to build a neural networks model that takes the motion measurements as input and predict the wheel wear. When the real wear damage data become available in the future, they can be used to update the neural networks model. Nevertheless, this neural networks model offers a potential to predict wheel wear directly from the motion data of the train. We must point out that it is expensive and time consuming to collect real wear data in reality and it is almost impossible to obtain the real wear data as a function of the motion of the train. Because of this, the proposed approach to use Archard's wear model as a surrogate to generate simulated wheel wear data is of high engineering value.

Neural networks has been extensively studied in many areas of research including remaining life estimation [139] and wear prediction [140] of mechanical systems. Neural networks has also been investigated for wheel and rail wear prediction. The wheel wear prediction using neural networks under different contact conditions is reported in [141]. A chaotic quantum particle swarm optimization based extreme learning machine algorithm is applied to the wheel tread profile optimization and tread wear prediction [142]. This section creates a new wheel wear neural networks model that uses motion measurements of the train to predict wheel wear. The main contributions of this section are summarized as below.

1. An extended state observer of a control system is for the first time used to predict wheel-rail interaction forces for the purpose of predicting wheel wear.

2. A data-driven algorithm is proposed to update the parameters of Archard's wear model by using motion measurements of the train.

3. Archard's wear model is then used as a surrogate to generate simulated wheel wear data to enable the development of a neural networks model of wheel wear with motion measurements as inputs. This approach has a potential to substantially expand our ability to accurately predict wheel wear and therefore to schedule maintenance in an optimal manner.

We adopt Archard's model to predict the wear damage, which describes the wheel-rail contact behavior. The parameters of Archard's model are determined in the ideal laboratory environment and usually treated as constant. However, wheel-rail contact is a complex nonlinear time-varying process during the train operation. This suggests that the parameters of Archard's model may also be time-varying and different from the nominal values determined in the laboratory setting. In this section, a recursive least squares algorithm is adopted to update the contact model parameters by making use of real-time motion measurements, *when such new data become available in the real application.*

6.3.1 Wheel-rail Interactions

Recall the bogie model in Equation (6.17), the top four lines involving the wheel-rail contact forces and moments are explicitly listed below,

$$\begin{aligned} m_w\ddot{y}_{w1} - k_{py}\dot{y}_{w1} - bk_{py}\dot{\varphi}_{w1} + k_{py}y_{w1} &= -F_{la1} \\ I_w\ddot{\varphi}_{w1} - l_1^2k_{px}\dot{\varphi}_{w1} + l_1^2k_{px}\varphi_{w1} &= -M_{la1} \\ m_w\ddot{y}_{w2} - k_{py}\dot{y}_{w2} - bk_{py}\dot{\varphi}_{w2} + k_{py}y_{w2} &= -F_{la2} \\ I_w\ddot{\varphi}_{w2} - l_1^2k_{px}\dot{\varphi}_{w2} + l_1^2k_{px}\varphi_{w2} &= -M_{la2} \end{aligned} \tag{6.23}$$

where F_{la1}, M_{la1}, F_{la2} and M_{la2} are the wheel-rail contact forces and moments, and can be expressed as,

$$\begin{aligned} F_{la1} &= \frac{2f_\eta}{v}\dot{y}_{w1} - 2f_\eta\varphi_{w1} + k_{gy}y_{w1} \\ M_{la1} &= \frac{2l_0^2f_\xi}{v}\dot{\varphi}_{w1} + \frac{2\lambda_e l_0 f_\xi}{r_0}y_{w1} + k_{g\psi}\varphi_{w1} \\ F_{la2} &= \frac{2f_\eta}{v}\dot{y}_{w2} - 2f_\eta\varphi_{w2} + k_{gy}y_{w2} \\ M_{la2} &= \frac{2l_0^2f_\xi}{v}\dot{\varphi}_{w2} + \frac{2\lambda_e l_0 f_\xi}{r_0}y_{w2} + k_{g\psi}\varphi_{w2} \end{aligned} \tag{6.24}$$

In reference [33], these terms appear in the damping and stiffness matrices as asymmetrical elements and are responsible for hunting instability as the train speed increases. In this work, we explicitly state their nature, i.e. the wheel-rail contact forces and moments

Define a vector $\mathbf{F}_{un} = [F_{la1}, M_{la1}, F_{la2}, M_{la2}]^T$. $\mathbf{F}_{un}$ is related to $\mathbf{f}(t)$ in Equation (6.18) as follows,

$$\mathbf{F}_{la}(t) = \begin{bmatrix} \mathbf{F}_{un} \\ \mathbf{0}_{4\times1} \end{bmatrix} = -\mathbf{M}\begin{bmatrix} \mathbf{f}(t) \\ \mathbf{0}_{4\times1} \end{bmatrix} \tag{6.25}$$

The parameters in Equation (6.24) include the lateral creep coefficient f_η, longitudinal creep coefficient f_ξ, gravitational stiffness k_{gy}. f_η describes the contact between the wheel and rail in the lateral direction. f_ξ describes the wheel-rail contact in the longitudinal direction. k_{gy} is related to the normal force at the wheel-rail contact point. These three parameters are usually determined in the laboratory and can be updated if and when the new data is available to improve the accuracy of wear prediction. A recursive least squares (RLS) algorithm is adopted for updating the parameters.

Let $F_{la1}(i)$ and $M_{la1}(i)$ denote their values at the i^{th} sample time step. $\dot{y}_{w1}(i)$, $y_{w1}(i)$, $\dot{\varphi}_{w1}(i)$ and $\varphi_{w1}(i)$ are the motion measurements at the same time. Consider identification of the three parameters f_η, f_ξ and k_{gy}. By definition, we have

$$\begin{bmatrix} F_{la1}(i) \\ M_{la1}(i) \end{bmatrix} = \begin{bmatrix} \left(\frac{2\dot{y}_{w1}(i)}{v} - 2\varphi_{w1}(i)\right) f_\eta + k_{gy} y_{w1}(i) \\ (\frac{2l_0^2}{v}\dot{\varphi}_{w1}(i) + \frac{2\lambda_e l_0}{r_0} y_{w1}(i)) f_\xi + k_{g\psi}\varphi_{w1}(i) \end{bmatrix} \quad \text{or}$$

$$\mathbf{y}_{la1}(i) = \boldsymbol{\varphi}(i)\boldsymbol{\eta} \tag{6.26}$$

where

$$\mathbf{y}_{la1}(i) = \begin{bmatrix} F_{la1}(i) \\ M_{la1}(i) \end{bmatrix} \tag{6.27}$$

$$\boldsymbol{\varphi}(i) = \begin{bmatrix} \frac{2\dot{y}_{w1}(i)}{v} - 2\varphi_{w1}(i) & 0 & y_{w1}(i) \\ 0 & \frac{2l_0^2}{v}\dot{\varphi}_{w1}(i) + \frac{2\lambda_e l_0}{r_0} y_{w1}(i) & 0 \end{bmatrix} \tag{6.28}$$

$$\boldsymbol{\eta} = [f_\eta, f_\xi, k_{gy}]^T \tag{6.29}$$

$\mathbf{y}_{la1}(i) \in \mathbb{R}^{2\times1}$, $\boldsymbol{\varphi}(i) \in \mathbb{R}^{2\times3}$ and $\boldsymbol{\eta} \in \mathbb{R}^{3\times1}$.

Define an objective function as,

$$\begin{aligned} J &= \frac{1}{2}\sum_{i=1}^{n_s} ||\mathbf{y}_{la1}(i) - \boldsymbol{\varphi}(i)\hat{\boldsymbol{\eta}}||^2 \\ &= \frac{1}{2}\sum_{i=1}^{n_s} (\mathbf{y}_{la1}(i) - \boldsymbol{\varphi}(i)\hat{\boldsymbol{\eta}})^T(\mathbf{y}_{la1}(i) - \boldsymbol{\varphi}(i)\hat{\boldsymbol{\eta}}) \\ &= \frac{1}{2}\sum_{i=1}^{n_s} [\mathbf{y}_{la1}^T(i)\mathbf{y}_{la1}(i) - 2\mathbf{y}_{la1}^T(i)\boldsymbol{\varphi}(i)\hat{\boldsymbol{\eta}} + \hat{\boldsymbol{\eta}}^T\boldsymbol{\varphi}^T(i)\boldsymbol{\varphi}(i)\hat{\boldsymbol{\eta}}] \end{aligned}$$

where n_s is the number of sampled data points of the motion signals and $\hat{\boldsymbol{\eta}}$ is an estimate of the parameter vector $\boldsymbol{\eta}$. $\hat{\boldsymbol{\eta}}$ is determined to minimize the objective function J. The optimal parameters can be computed with the recursive least squares algorithm [100] as illustrated in Section 5.3. The final recursive equations are summarized as follows.

$$\hat{\boldsymbol{\eta}}(n_s) = \hat{\boldsymbol{\eta}}(n_s - 1) + \mathbf{K}(n_s)[\mathbf{y}_{la1}(n_s) - \boldsymbol{\varphi}(n_s)\hat{\boldsymbol{\eta}}(n_s - 1)] \tag{6.30}$$

Table 6.3: Comparison between the nominal and estimated parameters

Symbol	Nominal value	Estimated value	Percentage error
f_η	8624000	8590626	0.3870
f_ξ	8624000	8683803	0.3870
k_{gy}	27596	27742	0.2637

where

$$\begin{aligned}\mathbf{K}(n_s) &= \mathbf{P}(n_s)\boldsymbol{\varphi}^T(n_s) \qquad (6.31)\\ &= \mathbf{P}(n_s-1)\boldsymbol{\varphi}^T(n_s)[\mathbf{I}+\boldsymbol{\varphi}(n_s)\mathbf{P}(n_s-1)\boldsymbol{\varphi}^T(n_s)]^{-1}\\ \mathbf{P}(n_s) &= [\mathbf{I}-\mathbf{K}(n_s)\boldsymbol{\varphi}(n_s)]\mathbf{P}(n_s-1) \qquad (6.32)\end{aligned}$$

Some simulation results are presented below. Figure 6.18 shows the convergence of parameter estimation with the RLS algorithm. As an example, the predicted lateral force $\hat{F}_{la1}$ and its error are shown in Figure 6.19. Recall that once the parameters are determined, the normal force N between the wheel and rail can be obtained from Equation (6.39). Table 6.3 summarizes the percentage error of estimation. The small error strongly supports the validity of the proposed parameter estimation approach.

A remark on the computational setting is in order. The time history of simulated motion measurements is about $t \approx 10s$ long with a sampling rate $0.0001s$. After discarding the transient responses, we use 2000 data points to obtain the initial estimate in Equation (5.30).

In the simulations, the parameters f_η, f_ξ and k_{gy} are assumed to be constant. When they are treated as slowly time-varying, as is the case during the train operation, the exponential forgetting RLS can be implemented to deal with time-varying nature of the parameters.

In the data of bogie motion, some terms such as $k_{g\psi}\varphi_{w1}$ and $k_{g\psi}\varphi_{w2}$ in Equation (6.24) have smaller values than others by several orders of magnitude. The parameters in those terms can be difficult to accurately identify with the RLS algorithm from noisy measurements because of the low signal-to-noise ratio. Since those terms are small and influence wheel wear to a lesser extend, their nominal parameters can be used for wear studies without the risk of significant error.

6.3.2 Wheel Wear Estimation

In this section, we present the procedure to estimate wheel wear. We make use of Archard's wear model [133], which is a common model for wheel wear of rail vehicles. The estimated lateral force in Section 6.3.1 determines the wheel-rail normal force, which is used in Archard's wear model.

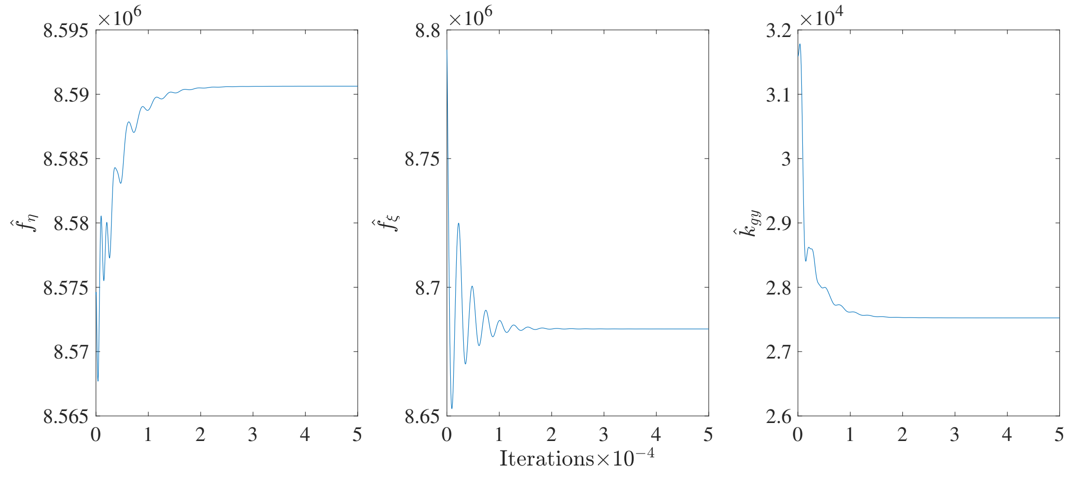


Figure 6.18: Estimation convergence of unknown parameters.

The lateral force is first estimated from MATLAB Simulink with 30 km running distance and then applied in Archard's model for wheel wear depth prediction. We first introduce Archard's wear model.

6.3.2.1 Wear Model

As discussed in the introduction, we adopt Archard's wear model in [136–138]. It is defined as,

$$W = \frac{KNs}{H} \tag{6.33}$$

where W (mm^3) is the total volume of wear. K is the dimensionless wear coefficient. N (Newton) is the total normal force acting on the wheel at the contact point, as shown in Figure 6.20. s (mm) is the sliding distance between the wheel and rail. H (MPa) is the hardness of the softest contacting surfaces. To estimate the wheel wear, we obtain the normal force at the contact point of wheel and rail from the estimated lateral force on the wheel.

Recall the equation of motion for bogie system in the lateral direction,

$$m_w\ddot{y}_{w1} + \frac{2f_\eta}{v}\dot{y}_{w1} + k_{py}y_{w1} + k_{gy}y_{w1} - 2f_\eta\varphi_{w1} - k_{py}\dot{y}_{w1} - bk_{py}\dot{\varphi}_{w1} = 0 \tag{6.34}$$

The equation contains lateral creep forces and gravitational stiffness effect. The combination of these terms are estimated by the ESO as discussed before. The lateral creep force is defined as,

$$F_{creep} = \frac{2f_\eta}{v}\dot{y}_{w1} - 2f_\eta\varphi_{w1} \tag{6.35}$$

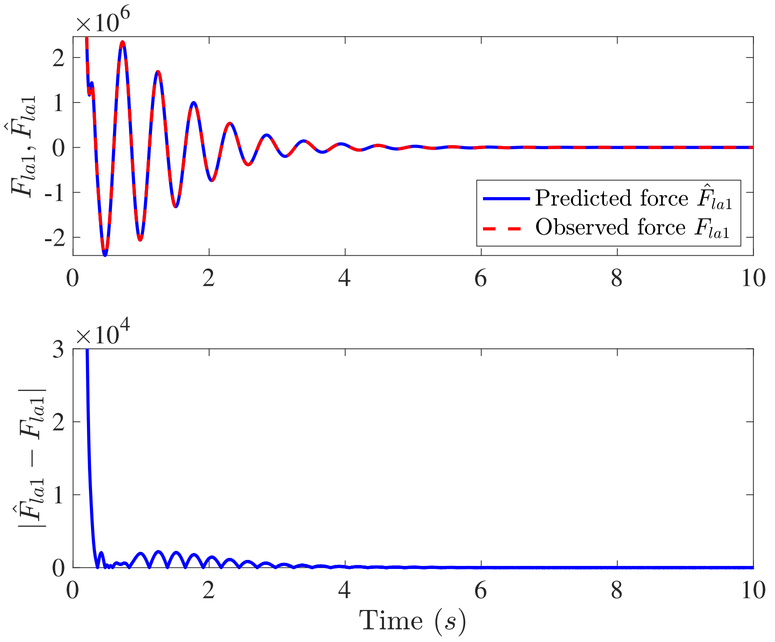


Figure 6.19: Top: Comparison between the predicted lateral force $\hat{F}_{la1}$ and the observed lateral force F_{la1}. Bottom: Prediction error of the lateral force.

The gravitational stiffness effect is given by,

$$F_{gravity} = k_{gy} y_{w1} \tag{6.36}$$

The estimated lateral force of a single wheelset of the bogie is given by,

$$F_{la1} = F_{creep} + F_{gravity} = \frac{2f_\eta}{v}\dot{y}_{w1} - 2f_\eta \varphi_{w1} + k_{gy} y_{w1} \tag{6.37}$$

The relationship between the normal force and the gravitational stiffness effect is given by,

$$N \sin\beta = k_{gy} y_{w1} \tag{6.38}$$

From Equations (6.37) and (6.38), the normal force can be obtained in terms of the estimated lateral force and creep force,

$$N = \frac{F_{la1} + 2f_\eta \varphi_{w1} - \frac{2f_\eta}{v}\dot{y}_{w1}}{\sin\beta} = \frac{k_{gy} y_{w1}}{\sin\beta} \tag{6.39}$$

where N is the normal force at the contact area, F_{la1} is the lateral force acting on a single wheel from the estimated force $\mathbf{F}_{la}$ in Section 6.3.1, and β is the contact

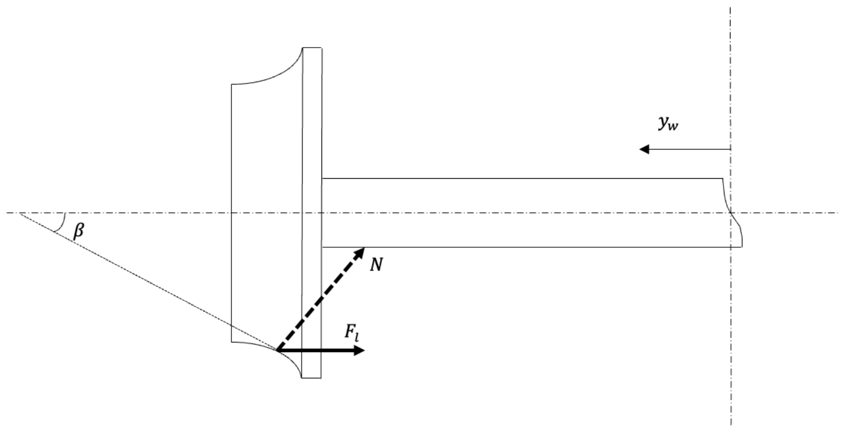


Figure 6.20: Contact force diagram between the wheel and rail.

angle between the wheel and rail. y_{w1}, φ_{w1} and $\dot{y}_{w1}$ are the motion signals, which are either directly measured by sensors or estimated from the ESO. When f_η are k_{gy} are known, the normal force can be obtained from Equation (6.39).

Note that the gravitational force is not drawn in Figure 6.20. However, its effect is included in the estimated lateral force. The contact angle β at flange is in a range from 65° to 70° [143]. In this study the contact angle is taken to be 70°. The softest material in the wheel and rail contact is usually the wheel. E8N wheel steel is considered because it is commonly used for train wheels in China and Europe [144]. The hardness of E8N steel is chosen as $H = 290$ (HV) $= 290 \times 9.807$ $= 2844.03$ (MPa) [145].

The wear coefficient K can be obtained from laboratory measurements [136]. The coefficient K in wheel-rail contact problem is determined by the sliding velocity and contact pressure between the wheel and rail, as illustrated by the chart in Figure 6.21. The wear coefficient is chosen based on the calculated sliding velocity $\mathbf{v}_{slip}$. The rigid slip distance s between wheel and rail is given by,

$$s = |\mathbf{v}_{slip}|\frac{\Delta x}{v} \tag{6.40}$$

where $\mathbf{v}_{slip}$ is the sliding velocity, Δx is length of contact along the longitudinal axes, v is the velocity of the wheel and is the same as the speed of the train in this study.

The sliding velocity $\mathbf{v}_{slip}$ can be calculated by making use of Kalker's theory. Kalker found that the rigid slip velocity between wheel and rail could be expressed

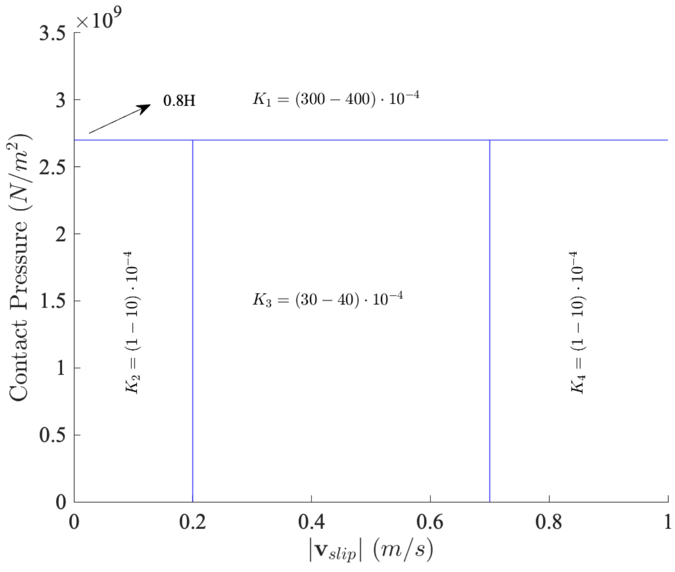


Figure 6.21: Archard map of the wear coefficient K as a function of sliding velocity and contact pressure.

as [146],

$$\mathbf{v}_{slip} = v[(\xi_x - \xi_{sp}y)\boldsymbol{i} + (\xi_y + \xi_{sp}x)\boldsymbol{j}] - v\frac{\partial \mathbf{u}_d}{\partial x} + \frac{\partial \mathbf{u}_d}{\partial t} \tag{6.41}$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ are unit vectors in the x and y directions, $\mathbf{u}_d$ is the displacement difference vector between wheel and rail in the $x-y$ plane. Consider steady state rolling and neglect the influence of elastic part. The rigid slip velocity can be further simplified as

$$\mathbf{v}_{slip} = v[(\xi_x - \xi_{sp}y)\boldsymbol{i} + (\xi_y + \xi_{sp}x)\boldsymbol{j}] \tag{6.42}$$

where ξ_x, ξ_y and ξ_{sp} are the longitudinal, lateral and spin creepages. x and y are the Cartesian coordinates of the contact area. The longitudinal, lateral and spin creepages are given by [147],

$$\begin{aligned}
\xi_x &= \frac{\gamma y_{w1}}{R_1} + \frac{D_c}{2}\frac{\dot{\varphi}}{v} \\
\xi_y &= \frac{\dot{y}_{w1}}{v} - \varphi \\
\xi_{sp} &= -\frac{\varphi}{v}\cos\gamma + \frac{1}{R_1}\sin\gamma
\end{aligned} \tag{6.43}$$

where φ is the yaw angle, R_1 is the principal rolling radius of the wheel, y_{w1} is the lateral displacement, γ is the wheel conicity and D_c is the distance between the contact spots. The creepages can be calculated by making use of the available bogie parameters and the estimated data from the observer. Once these creepages are available, we are ready to determine the sliding velocity with the wheel-rail contact area. The shape and size of the contact area between the wheel and rail are determined by Hertz's static solution. The shape of the contact region is assumed to be an ellipse. The area of the ellipse contact region is described by the semi-axes a and b,

$$a = m_h \left(\frac{3}{2} N \frac{1-\nu^2}{E} \frac{1}{A+B} \right)^{\frac{1}{3}} \tag{6.44}$$

$$b = n_h \left(\frac{3}{2} N \frac{1-\nu^2}{E} \frac{1}{A+B} \right)^{\frac{1}{3}} \tag{6.45}$$

where

$$A = \frac{1}{2R_1}, \quad B = \frac{1}{2} \left(\frac{1}{R_1'} + \frac{1}{R_2'} \right) \tag{6.46}$$

where R_1', R_2' are the principal transverse radii of curvature of the wheel and rail at contact point. ν is Poisson's ratio of the wheel, E is Young's modulus of elasticity of the wheel. m_h and n_h are Hertzian dimensionless coefficients related to the angle θ defined by [147],

$$\theta = \cos^{-1} \frac{|B-A|}{B+A} \tag{6.47}$$

Table 6.4 lists the coefficients m_h and n_h for various angles θ.

Table 6.4: Hertz coefficients m_h and n_h.

$\theta°$	m_h	n_h	$\theta°$	m_h	n_h	$\theta°$	m_h	n_h
10	6.604	0.3112	40	2.136	0.567	70	1.284	0.802
20	3.813	1.4123	50	1.754	0.641	80	1.128	0.893
30	2.731	0.493	60	1.486	0.717	90	1.000	1.000

Once the contact area is determined, Hertz's theory also gives the normal pressure acting on the wheel contact area [146],

$$P(x,y) = \frac{3N}{2\pi ab} \sqrt{1 - \left(\frac{x}{a}\right)^2 - \left(\frac{y}{b}\right)^2} \tag{6.48}$$

To compute the wear using Archard's wear model in Equation (6.33), we discretize the contact area into $n_c = 50$ small elements. Assume that the pressure is constant over the small element. Let (x_i, y_i) denote the center of the element and

$N_i = P(x_i, y_i)$ denote the pressure on the element ($i = 1, 2, \cdots, n_c$). According to Equation (6.33), the wear in each element is obtained as,

$$\Delta w_i = \frac{K N_i s}{H} \tag{6.49}$$

The average wear in the sliding region is given by,

$$\Delta \bar{w} = \frac{1}{n_c} \sum_{i=1}^{n_c} \Delta w_i \tag{6.50}$$

All the parameters for the wear study of the wheel and rail are listed in Table 6.5.

Figure 6.22 shows the wheel wear prediction over a travel distance 30 km. We can see that with increasing distance, more wheel wear is accumulated. The proposed method clearly can estimate the wheel wear over longer travel distance from the data of train motion measurements. Note that all the parameters of bogie system and wheel-rail contact parameters are referred from [2, 143–145, 147], other complicated wear model or different parameters can also be considered in the algorithm to predict the wheel-rail wear. Here we use the linear Archard's model [133] and a simplified bogie system from [2] to illustrate the potential of the algorithm. A flowchart is shown in Figure 6.23 to illustrate the overall wear estimation algorithm.

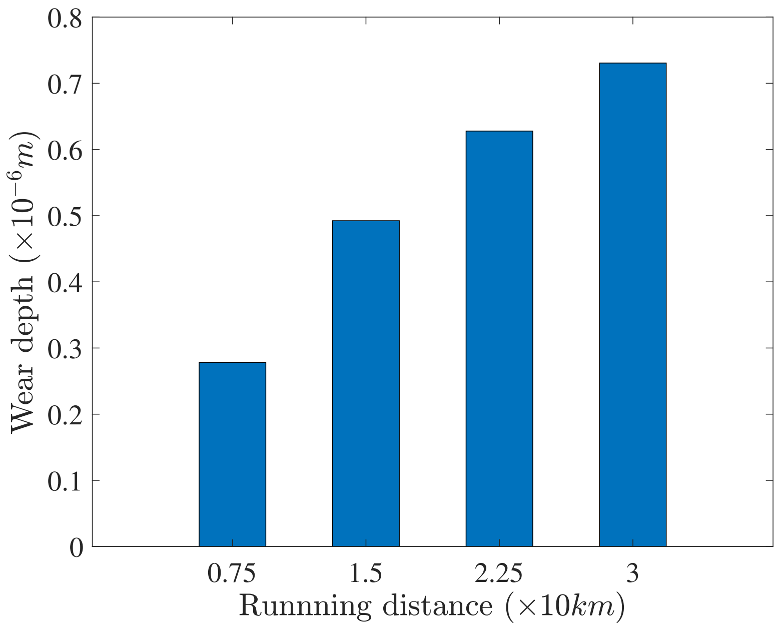


Figure 6.22: Accumulated wheel wear in the travel distance $30km$.

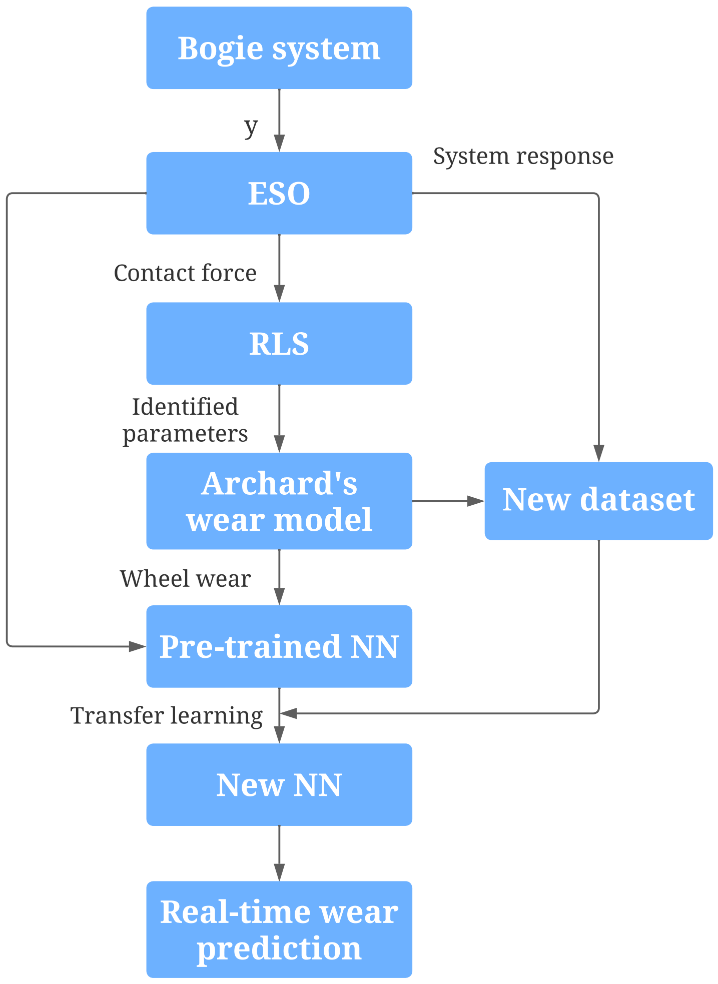


Figure 6.23: Flowchart of the wear estimation algorithm.

Table 6.5: Parameters of the wheel wear.

Symbol	Description	Value
W	Weight of the bogie	1639 kg
D_c	Distance of the contact spot	1.493 m
γ	Wheel rail conicity	0.2
ν	Poisson's ratio	0.3
E	Young's modulus of elasticity	2.1E11 N/m^2
R_1	Principal rolling radius of the wheel	0.625 m
R'_1	Principal rolling radius of the wheel	2.56 m
R'_2	Principal rolling radius of the wheel	0.25 m
H	Hardness of the wheel	290 HV
β	Contact angle	70°
K	Wear coefficient	-
m_h	Hertz coefficient	-
n_h	Hertz coefficient	-

6.3.3 Neural Networks based Wear Prediction

The previous model-based study of wheel wear establishes the fact that motion measurements of the train contain the information that determines wheel wear. In other words, wheel wear is a highly nonlinear and complex function of the motion of the train, as suggested by Archard's wear model. The question we would like to address next is: can we estimate wheel wear directly from motion measurements. To this end, we explore the use of neural networks, which is known to be able to model highly nonlinear and complex input-output relationships.

We treat the previous wear model as a surrogate to generate wear data for training the neural networks. The neural networks model takes motion measurements as input and outputs wear prediction. This is a valuable approach because wheel wear measurements are rare and cannot be obtained in real time. Once neural networks is trained, it can be used to better schedule wheel wear inspection and maintenance.

6.3.3.1 Neural Networks Model

We consider a deep neural networks to model the relationship between wheel wear and motion measurements of the bogie. It should be pointed out that each wheel can have a model. In the following, we consider wheelset #1 only. The inputs of the neural networks model include lateral displacement y_{w1}, its velocity $\dot{y}_{w1}$ and yaw angular velocity $\dot{\varphi}_{w1}$. The output is the wear depth $\Delta\bar{w}$ of wheelset #1. This selection of input and output for the neural networks model is based on the observation of Archard's wear model. The neural networks model establishes a nonlinear functional relationship denoted as,

$$\Delta\bar{w} = NN(y_{w1}, \dot{y}_{w1}, \dot{\varphi}_{w1}) \tag{6.51}$$

The structure of the neural networks model is shown in Figure 6.24. There are five hidden layers in the neural networks with 50 neurons for each layer. The activation function is chosen as the hyperbolic tangent function $\tanh(x)$. There are many other activation functions available in the neural networks library, including the sigmoid function, ReLU and Leaky ReLU. The choice of the number of hidden layers and neurons is made based on numerical experiments with the data. The neural networks used in this study is intended to demonstrate the feasibility of the proposed method and is not optimized.

The Adam optimization algorithm is used to train the neural networks [148]. Adam is a well studied stochastic gradient descent algorithm with many advantages, including high computational efficiency and available Python implementation in TensorFlow. The loss function of neural networks is the mean squared error of wear prediction defined as,

$$J = \frac{1}{2}(\Delta\bar{w}_{data} - \Delta\bar{w}_{predict})^2 \tag{6.52}$$

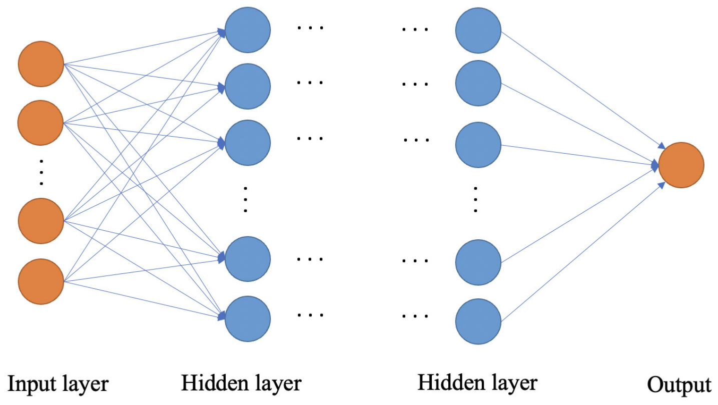


Figure 6.24: Structure of the neural networks model.

where $\Delta\bar{w}_{data}$ is the simulated wear depth from Archard's wear model, $\Delta\bar{w}_{predict}$ is the predicted wear depth by the neural networks model. An early stopping criterion is imposed to avoid over-fitting. That is to say, if the monitored loss function J does not decrease in consecutive 50 epochs, training stops.

The training data is generated by simulating the motion of the bogie at different operating velocities, such as 200 km/h, 250 km/h and 300 km/h. There are 8322 simulated input-output datasets. Pre-processing is crucial for training neural networks. All the input and output data are scaled to the range of [0,1]. 75 percent of the data is for training while the remaining 25 percent is for validation.

A sample history of the loss function is shown in Figure 6.25. The prediction of a test sample for the purpose of validation is shown in Figure 6.26. The corresponding predication error is shown in Figure 6.27. These figures show that the order of magnitude of the loss function converges to 10^{-4}, while the prediction of the neural networks on the test data is highly accurate. The training is done on a MacBook Air in 59.87 seconds.

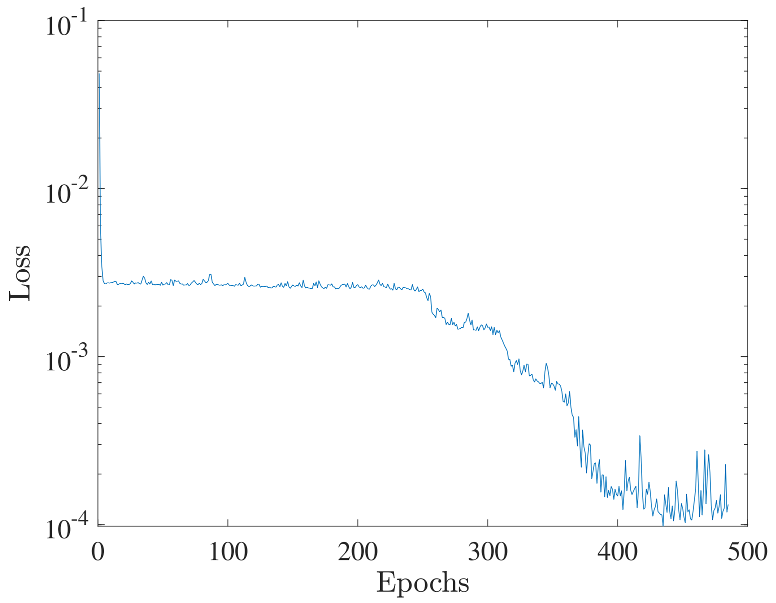


Figure 6.25: Loss function of the neural networks in training.

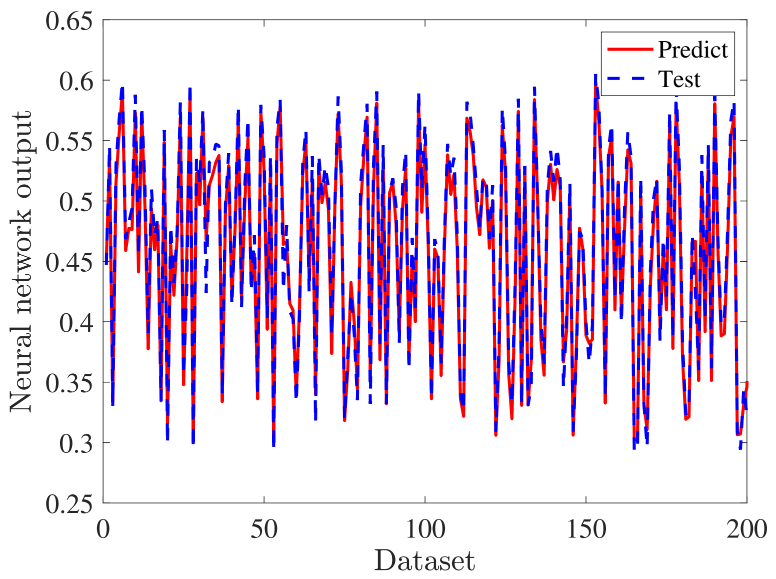


Figure 6.26: Validation of the neural networks on the test data.

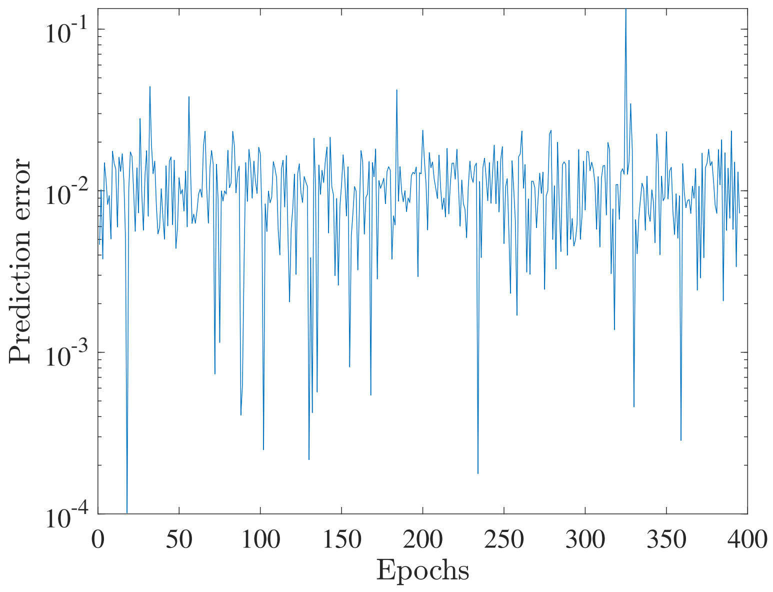


Figure 6.27: Prediction error of the neural networks on the test data.

6.3.3.2 Transfer Learning

The trained neural network in Section 6.3.3.1 gives good prediction over the test dataset. Note that the test data are part of the original dataset and related to the training data to some extend. During the operation of the train, the data collected in real time are influenced by environmental and operating conditions. The prediction of the trained neural networks with the real data collected in different conditions may not be accurate enough. To improve the performance and generality of the neural networks model, we make use of the transfer learning concept to partially retrain the neural networks with the new data.

Transfer learning is a machine learning technique to accomplish new similar tasks with the trained neural networks model. Here, we adopt the fine tuning technique of transfer learning [84]. Assume that the neural networks model is well trained with the available data. In the transfer learning, we freeze the weights in the hidden layers of the neural networks, and fine-tune the weights in the last layer with the new data. For the wheel wear prediction of the train, the neural networks model is first trained with the data generated from a given initial condition. Fine-tuning is then applied to update the neural networks model with the data from different initial conditions. The training of transfer learning takes 12.80 seconds on the same MacBook.

The predictions of the pre-trained neural network and the fine-tuned neural network are shown in Figures 6.28 and 6.29. It is observed that the neural networks model with the fine-tuning technique delivers much improved prediction with the new data. In fact, the re-trained neural networks model reaches a comprise between the new data and the old data, as can be seen by comparing Figures 6.26, 6.28, 6.29 and 6.30.

6.3.3.3 Wear Distribution

With the trained neural networks model, it is possible to study the influence of train motions on the wheel wear depth $\Delta\bar{w}$. In Archard's wear model, we can view inputs y_{w1}, $\dot{y}_{w1}$ and $\dot{\varphi}_{w1}$ as independent time-varying parameters. The wear can then be predicted with the help of the neural networks model in a range of possible motions. That is to say, we can obtain the distribution of the wear in the space of the possible motions and discover the hot spots in the space where certain combinations of the motions may cause the most wear. The results of this finding are presented in a ternary wear diagram.

In the ternary wear diagram, the inputs are normalized in the range $[0, 1]$. The normalized inputs are further scaled such that their sum is either one or 100% [149]. The ternary wear diagram is shown in Figure 6.31, where $\bar{y}_{w1}$, $\dot{\bar{y}}_{w1}$ and $\dot{\bar{\varphi}}_{w1}$ represents the normalized inputs of the neural networks model.

The yellow region of high wear value in Figure 6.31 indicates that the combinations of middle level lateral displacement y_{w1} and rotational speed $\dot{\varphi}_{w1}$ together

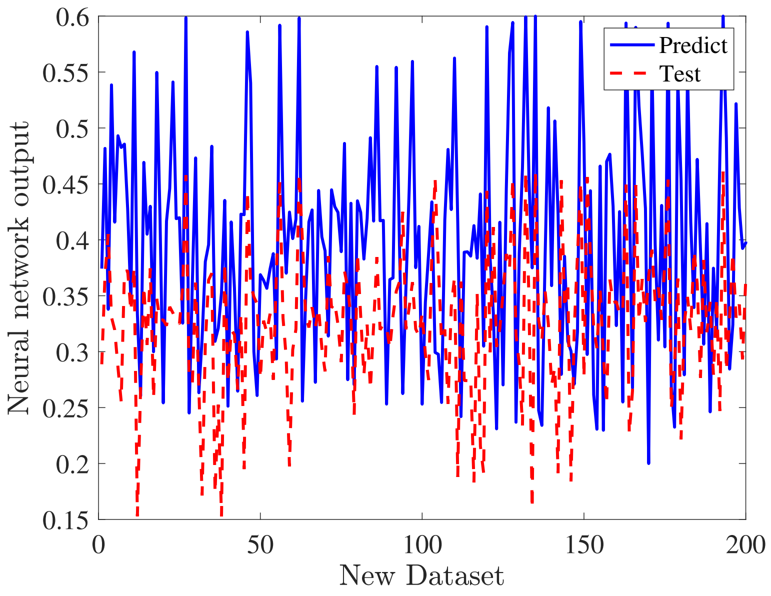


Figure 6.28: Validation of the original neural networks on the new test data.

with relatively smaller lateral velocity $\dot{y}_{w1}$ result in larger wheel wear depth. y_{w1} appears to be positively correlated to the value of wear depth. $\dot{y}_{w1}$ has a negative correlation to the wear depth. The wear appears to reach maximum in a middle range of $\dot{\varphi}_{w1}$ regardless the lateral displacement and its velocity.

6.3.4 Conclusions

In this section, we have presented a method to estimate the wheel wear from motion measurements of the train. The extended state observer of a control system is used to estimate wheel-rail interaction forces, which help to determine all the force terms in Archard's wear model. A parameter updating algorithm is developed to update the wear and creep parameters of Archard's model, thus making it more accurate for the actual train. The updated Archard's wear model is then used as a surrogate to simulate a large dataset of wheel wear from extensive complex train motions. A neural networks model is then developed to predict the wear as output while using simulated motions as inputs. A transfer learning technique is used to fine-tune the neural networks model when new data is fed to it, leading to a comprise of the neural networks model between the early training data and the new data. An example of application of the neural networks model is presented to discover the effect of train motions on the wear. The neural networks model has a potential to help engineers to develop effective and optimal schedule for inspection and maintenance of high-speed train wheels for wear damages.

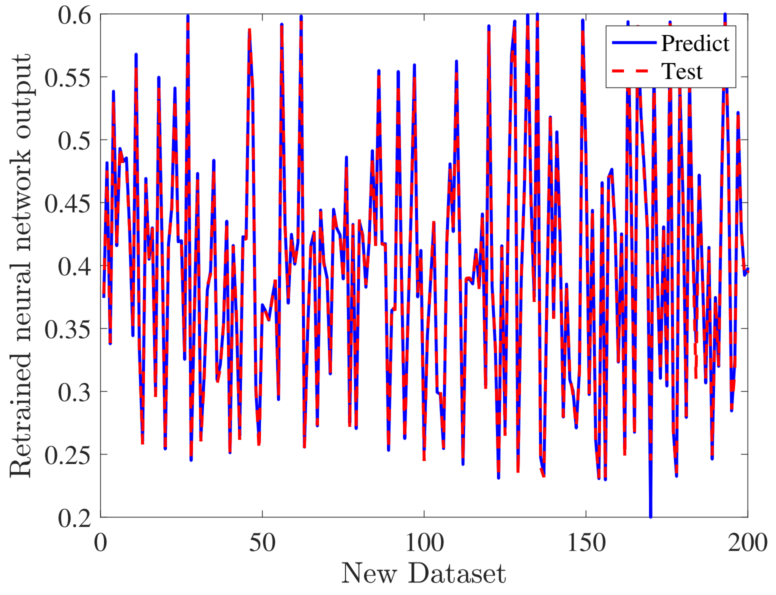


Figure 6.29: Validation of the retrained neural networks on the new test data.

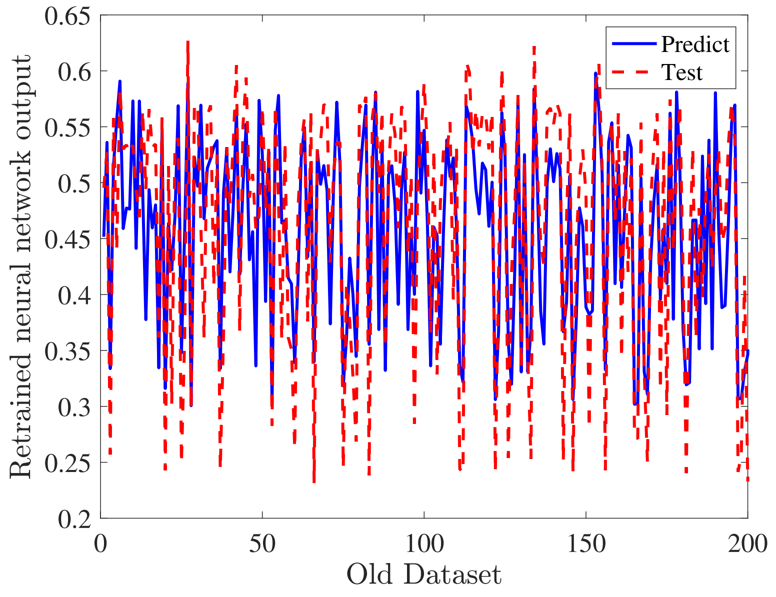


Figure 6.30: Validation of the retrained neural networks on the old test data. This is to be compared with Figure 6.26.

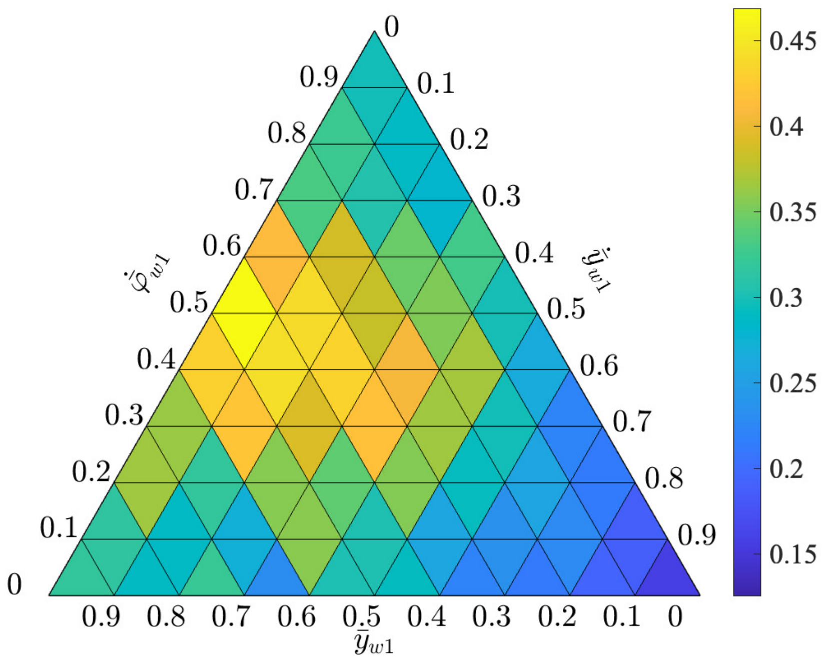


Figure 6.31: Wheel wear as a function of train motions (y_{w1},$\dot{y}_{w1}$,$\dot{\varphi}_{w1}$) based on predictions of the neural network model. The original ranges of ternary diagram axes are: $y_{w1} \in [0.0024, 0.0095]$, $\dot{y}_{w1} \in [0.0017, 0.3119]$, $\dot{\varphi}_{w1} \in [-0.4379, 0.1364]$.

Chapter 7
ENGINEERING APPLICATION II

A Delta robot is an over-actuated parallel robot composed of three arms connected to an end effector mounted on the base. Delta robots are known for their high speed, accuracy, and versatility, making them suitable for various tasks such as assembly, pick-and-place, and classification. One key aspect of Delta robot control is the dynamic model, which describes the relationship between the joint inputs and the position of the end effector. There are several approaches to modeling the Delta robot. One of the most popular approaches is Lagrangian dynamics [150–152]. Another common approach to modeling the Delta robot is to use inverse kinematics [153,154], with the help of the geometry of the robot's joints and links. Additionally, screw theory, which represents the robot's motion in terms of screw axes and twists, has also been adopted to model the dynamics of the Delta robot [155]. The dynamic model is essential for motion planning and trajectory tracking, as it allows the controller to calculate the required joint inputs to achieve a desired end effector position.

Delta robot has a highly nonlinear complicated dynamic model, making its control design challenging. Plenty of nonlinear and adaptive control algorithms have been implemented for the trajectory tracking and disturbance rejection of the Delta robot. Sliding mode control is one of the most popular nonlinear control algorithms. It has been extensively applied to the trajectory tracking of Delta robot combined with fuzzy neural network [156], nonlinear proportional-derivative (PD) control [157], and synergetic control [158]. RBFNN were adopted in [158] to compensate for the unknown disturbances. Ref. [159] proposed an online estimation approach to compensate for various uncertainties in the Delta robot. An adaptive active disturbance rejection control was adopted for the output-based robust trajectory tracking of the Delta robot in [150]. Iterative learning control has also been applied to the trajectory tracking of the Delta robot with high performance. Ref. [160] proposed a PD-type ILC combined with a PD controller to improve the iteration performance. Model-free iterative learning control was further implemented on the Delta robot with nonrepetitive trajectories [161]. An online estimation approach was proposed to compensate for the uncertainties to improve the tracking performance in [159]. Moreover, a model reference adaptive control has also been adopted for control of the Delta robot. Combining an identified linear model with the mode reference adaptive control has shown a significant performance improvement compared to the three commonly used methods: PID, adaptive control, and sliding mode control algorithms [162].

Most of the advanced control algorithms mentioned above are either data-driven or model based. Model-based control algorithms are highly dependent on the accuracy of the model. Therefore, they have high requirements on the control robustness in terms of the model uncertainties and disturbances. On the other hand, data-driven algorithms have no requirements for the accuracy of the model. However, the safety issue needs to be considered carefully during the operation. Therefore, the robustness and performance of trajectory tracking are significant challenges in the control design of the Delta robot. Moreover, another challenge in Delta robot control is the singularity, which occurs when the end effector is positioned at specific locations where the kinematic model degenerates and the joint angles become indeterminate, which can lead to erratic behavior and potential damage to the robot. Therefore, understanding the robot's geometry and workspace is crucial to avoid singularities, which can cause significant problems during operation.

In this study, a Delta robot is designed with stepper motors as inputs. Once the desired trajectory of the end effector is determined, inverse kinematics can be adopted to solve for the joint inputs analytically or numerically using optimization methods. Moreover, control algorithms can be implemented to further improve the motion reliability and the trajectory tracking accuracy of the Delta robot. Instead of using optimal control, this section proposes a hybrid of the data-driven and model-based sliding mode control algorithm to improve its trajectory tracking performance. The inverse kinematics of the Delta robot is analytically difficult to be expressed in the control-affine form, which complicates the control design process. Here, a neural network structure is proposed to approximate the dynamic model of the Delta robot, considering the stepper motor angles and velocities as inputs. Furthermore, the model-based sliding mode control is adopted for trajectory tracking. A data-driven neural networks algorithm that makes full use of the inverse kinematics is developed to approximate the dynamic model of the system. Sliding mode control is implemented to ensure the trajectory tracking performance of the Delta robot.

Section 7.1 presents the dynamic model approximation using neural networks. Section 7.2 presents the sliding mode control with neural networks approximated model. Section 7.3 concludes the work.

7.1 Dynamic Model of Delta Robot

A mechanical strucutre of the three DOF Delta robot is shown in Figure 7.1. Figure 7.2 shows the geometry of the Delta robot in 3D. A typical Delta robot is composed of a fixed platform, a moving platform, three active arms and three passive arms connected to an end effector mounted on the base. The active arms of the robot are driven by the rotation actuators, which are stepper motors in this study.

The parameters of the Delta robot are given in Table 7.1 and the specifications of components used for fabricating the Delta robot are shown in Table 7.2.

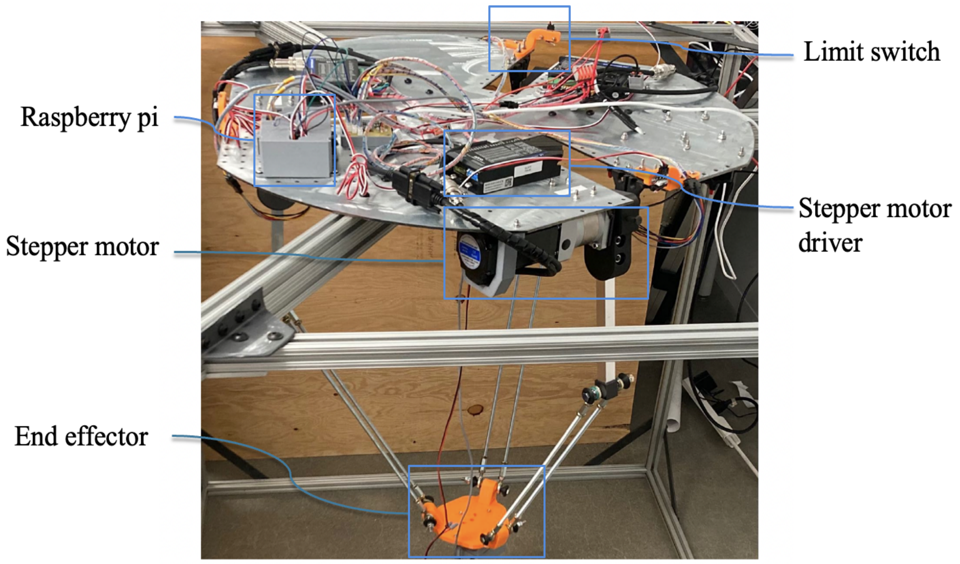


Figure 7.1: Mechanical structure of the Delta robot.

Table 7.1: Parameters of the Delta robot.

Description	Notation	Value
Radius of the fixed platform	R	0.325 m
Radius of the moving platform	r	0.075 m
Length of the active arm	r_f	0.5 m
Length of the passive arm	r_e	0.25 m
Mass of the active arm	m_f	0.205 kg
Mass of the passive arm	m_e	0.153 kg
Mass of the end effector	m_b	0.653 kg

Table 7.2: The specifications of the components utilized in the construction of the Delta robot.

Product	Model	Specification
Stepper Motor	23HS30-5004D-E1000	Motor type: Bipolar Holding torque: 2.00 N$\cdot m$ Step accuracy: $\pm 5\%$ Resistance: $0.42 \pm 10\%$ Inductance: $1.72 \pm 20\%$
Stepper Motor Driver	CL57T	Weight: 290 g Input voltage: 24-48 VDC Pulse input frequency: 0-500 kHz Min. Pulse width: $1 \mu S$
Planetary Gearbox	PLE23-G10-D8	Gear ratio: 10 Efficiency: 94.00% Max.Permissible Torque: 10 N$\cdot m$ Moment permissible torque: 20 N$\cdot m$ Backlash(arcmin): ≤ 15 Noise $\leq$ 60dB
Angle Sensor	AS5600	12-bit DAC output resolution I^2C interface
Laser Sensor	VL53L1X	50 Hz ranging frequency Field-of-View : 27° I^2C interface
Raspberry pi 4	Model B	64-bit Cortex-A72 processor 4GB LPDDR4 RAM

7.1.1 Inverse Kinematics

One of the most important algorithms of modeling the Delta robot is inverse kinematics. The inverse kinematics of the Delta robot can be solved geometrically by making use of the links and joints. From Figure 7.2, the relationship between the positions of the joints and the joint angles θ_i can be given as,

$$\theta_i = \arctan\left(\frac{Z_{Ji}}{Y_{Fi} - Y_{Ji}}\right) \tag{7.1}$$

where Y_{Fi} and Y_{Ji} are the positions of the point F_i and J_i in Y direction, Z_{Ji} is the position of the point J_i in Z direction. The positions of the joint Y_{Fi} and Y_{Ji} satisfy the geometry conditions,

$$\begin{cases} (Y_{Ji} - Y_{Fi})^2 + (Z_{Ji} - Z_{Fi})^2 = r_f^2 \\ (Y_{Ji} - Y_{E'i})^2 + (Z_{Ji} - Z_{E'i})^2 = r_e^2 - x_0^2 \end{cases} \tag{7.2}$$

Once the desired joint angles are obtained, one solution to compute the desired joint velocities is to discretize the desired joint angle $\theta_i(k\Delta t)$ at each discrete time step k, the joint velocities $\dot{\theta}_i$ is given as,

$$\dot{\theta}_i = (\theta_i(k\Delta t) - \theta_i((k-1)\Delta t))/\Delta t \tag{7.3}$$

in the time interval $[(k-1)\Delta t, k\Delta t]$. Another solution is to calculate the inverse velocity kinematics [163] that can be described by the Jacobian matrix J relates to the joint velocities $\dot{\boldsymbol{\theta}} = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$ and the desired end effector position velocities $\boldsymbol{\mathcal{V}} = [\dot{x}_0, \dot{y}_0, \dot{z}_0]^T$,

$$\dot{\boldsymbol{\theta}} = J(x_0, y_0, z_0)\boldsymbol{\mathcal{V}} \tag{7.4}$$

where (x_0, y_0, z_0) describes the position of the end effector at the centroid. From Equations (7.1) to (7.4), it is suggested that the desired joint velocities are nonlinear functions of the desired positions and its velocities.

In this study, the inputs from stepper motors $[\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}]$ are treated as control to the dynamic system. When using the discretization or Jacobian matrix method to compute joint velocities, it is typical for minor errors in the computed velocities to accumulate over time. These errors can lead to increasing position error, which can be problematic in the trajectory tracking control that requires high accuracy. Due to its complex and nonlinear structure, the equations of motion for the Delta robot are not clearly described by the inverse kinematics or inverse velocity kinematics. Therefore, it is essential to accurately model its dynamics to design efficient control algorithms for the Delta robot. One approach to solving this problem is using the approximation property of neural networks. Neural networks are a type of machine learning algorithm that can learn the underlying physics model between

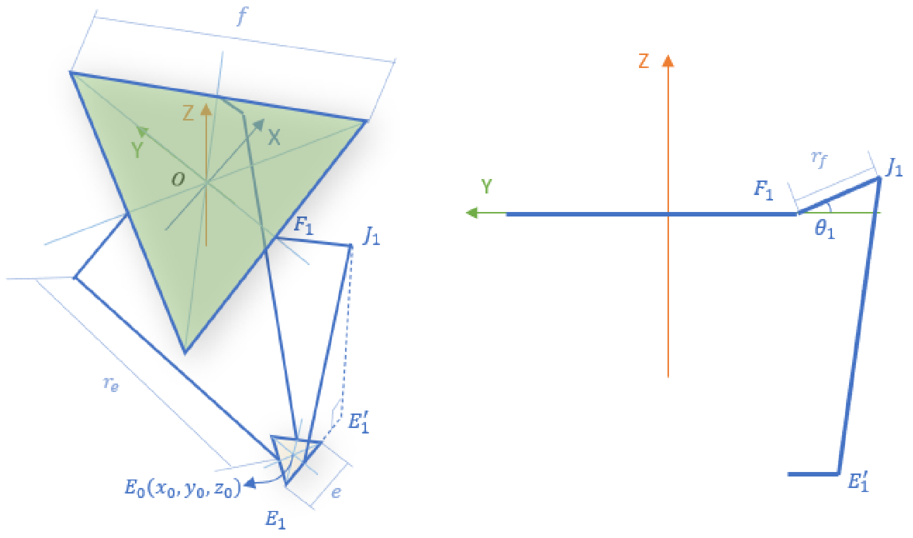


Figure 7.2: Figures illustrating the geometry for modeling the inverse kinematics of a Delta robot. Left: 3D geometry of the Delta robot. Right: 2D geometry of the Delta robot joints and links.

inputs and outputs. Training a neural network on input-output datasets obtained from the Delta robot makes it possible to accurately approximate the dynamics, especially when the experimental data is available. More details regarding to the neural network approximation of the Delta robot's dynamics can be found in the following section.

7.1.2 Neural Networks Model

In this section, the neural networks are proposed to approximate the dynamic model of the Delta robot.

Assumption 1. The dynamic model is assumed to be a nonlinear control-affine system with joint angles and velocities as inputs. Control-affine systems have a simpler structure that makes the control design relatively straightforward.

Consider the nonlinear control-affine model of Delta robot given as,

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{g}(\mathbf{x}, \dot{\mathbf{x}})\mathbf{u} \tag{7.5}$$

where $\mathbf{x} = [p_x, p_y, p_z]^T$, $\mathbf{u} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]^T$, $\mathbf{f}(\cdot) \in \mathbb{R}^{3\times 1}$ and $\mathbf{g}(\cdot) \in \mathbb{R}^{3\times 6}$ are nonlinear functions of their arguments, p_x, p_y, p_z represent the position of the end effector at the centroid along x, y and z axis. Deriving the equation of motion of the Delta robot in control-affine form with specific control inputs is challenging. Although several dynamic models are available for the Delta robot that uses the computed-torque approach, they are not applicable in this study since the inputs of the Delta robot are the joint angles and velocities. Therefore, an alternative approach needs to be used to model the Delta robot's dynamics. Here, the neural networks are adopted to approximate the dynamic model of the Delta robot designed in this study. Two neural networks are constructed to approximate the nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$, respectively. Notably, one neural network can also be apoted to approximate the two nonlinear functions with an extra hidden layer and an extra input. Finding the relationship between the control inputs and system states matters.

To cover the typical workspace of the Delta robot, the joint angles are randomly generated within a bounded range $\theta_i \in [-30^\circ, 120^\circ]$ as shown in Figure 7.3. These joint angles were inputs to collect $\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}$ and $\dot{\theta}_i$ from the SimScape model of Delta robot. The generated position points of the end effector are shown in Figure 7.4. This Figure shows the maximum workspace of the robot with parameters listed in Table 7.1. The inputs for neural networks are $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\mathbf{u}$. The output for the neural networks is $\hat{\ddot{\mathbf{x}}}$, which can be expressed in terms of the two neural networks $\hat{\mathbf{f}}(\mathbf{x}, \dot{\mathbf{x}})$ and $\hat{\mathbf{g}}(\mathbf{x}, \dot{\mathbf{x}})$,

$$\hat{\ddot{\mathbf{x}}}(i) = \hat{\mathbf{f}}(\mathbf{x}(i), \dot{\mathbf{x}}(i)) + \hat{\mathbf{g}}(\mathbf{x}(i), \dot{\mathbf{x}}(i))\mathbf{u} \tag{7.6}$$

where i is the index for the sampling point. The structure of neural networks is shown in Figure 7.5. The objective function is defined as,

$$\mathbf{J} = \frac{1}{2}\sum_{i=1}^{n_s}(\hat{\ddot{\mathbf{x}}}(i) - \ddot{\mathbf{x}}(i))^2 \tag{7.7}$$

where $\hat{\ddot{\mathbf{x}}}(i)$ is the prediction from neural networks, $\ddot{\mathbf{x}}(i)$ is the system response data collected from simulation or experimental, n_s is the total sampling points. Detailed information of the number of hidden layers, type of activation function and number of neurons can be found in Table 7.3. The number of training epochs is set to be 20000. The optimization algorithm chosen is stochastic gradient descent (SGD). Dropout with a frequency of 0.5 is adopted to prevent overfitting. The SGD algorithm [164] is one of the most popular stochastic optimization methods in the machine learning and deep learning community. It has been adopted as the optimization algorithm to train the neural networks in this study. The stochastic gradient descent maintains a single learning rate to update all weights compared to Adam [148].

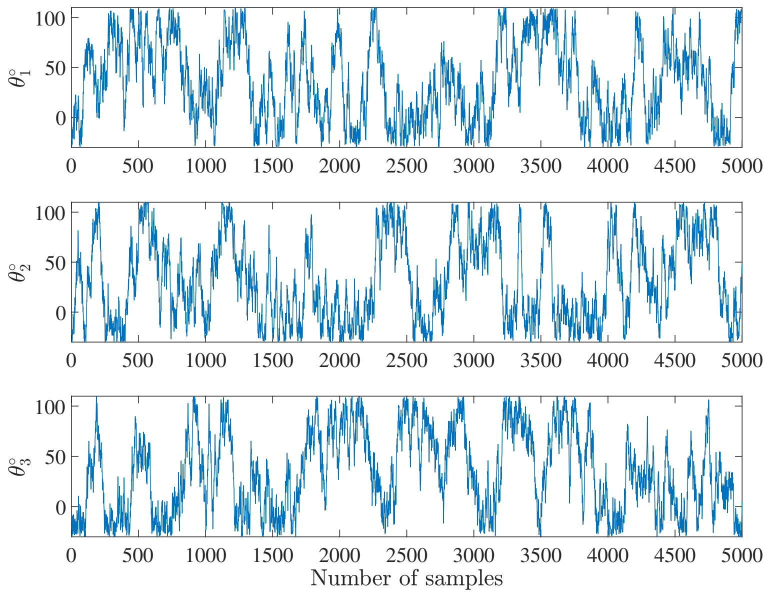


Figure 7.3: Random joint angles for SimScape Delta robot model. Top: Joint angle θ_1 for steppter motor 1. Middle: Joint angle θ_2 for stepper motor 2. Bottom: Joint angle θ_3 for stepper motor 3.

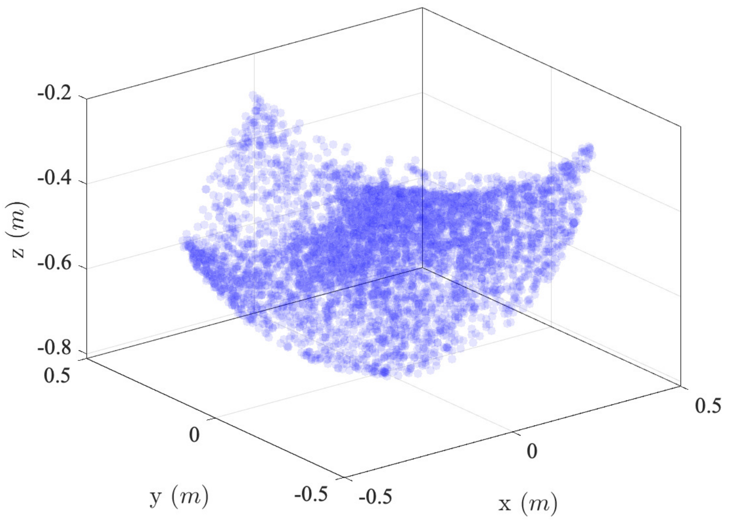


Figure 7.4: Random sampling points for Delta robot in 3D.

Table 7.3: Summary of neural networks.

Function	No. of hidden layers	Activation function	No. of neurons
$\hat{\mathbf{f}}(\mathbf{x})$	1	sigmoid	100
$\hat{\mathbf{g}}(\mathbf{x})$	1	sigmoid	100

7.2 Sliding Mode Control

Recall the control-affine nonlinear model in terms of the neural networks in Equation (7.5),

$$\ddot{\mathbf{x}} = \hat{\mathbf{f}}(\mathbf{x}, \dot{\mathbf{x}}) + \hat{\mathbf{g}}(\mathbf{x}, \dot{\mathbf{x}})\mathbf{u} \tag{7.8}$$

Assume that the Delta robot will track the desired trajectory $\mathbf{x}_d(t)$. The tracking error and its derivative are given as,

$$\mathbf{e}(t) = \mathbf{x}_d(t) - \mathbf{x}(t), \quad \dot{\mathbf{e}}(t) = \dot{\mathbf{x}}_d(t) - \dot{\mathbf{x}}(t) \tag{7.9}$$

Design the sliding surface as,

$$\mathbf{s}(t) = \mathbf{C}\mathbf{e}(t) + \dot{\mathbf{e}}(t) \tag{7.10}$$

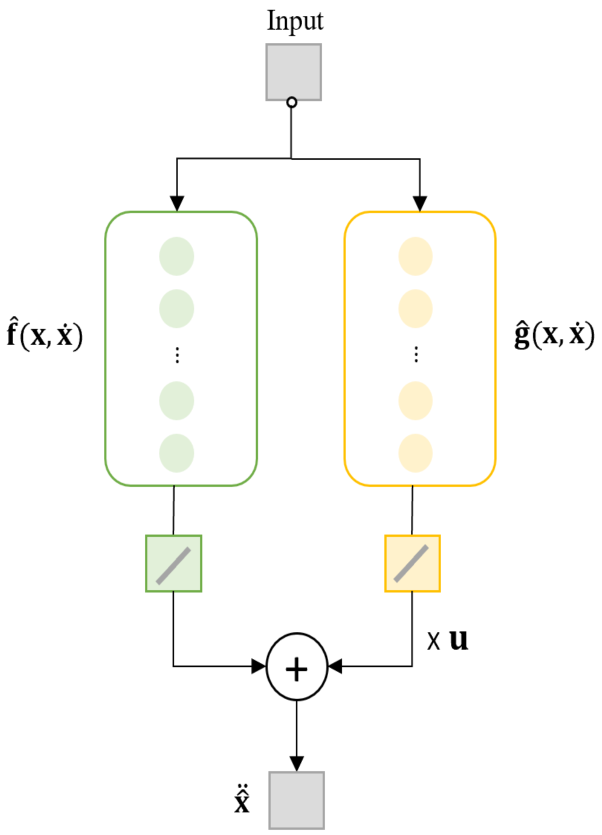
where $\mathbf{C} \in \mathbb{R}^{3\times 3}$ is positive definite and satisfies Hurwitz condition. Design the sliding mode control as [165]

$$\mathbf{u}(t) = \hat{\mathbf{g}}^{-1}\left(\gamma \tanh\left(\frac{\mathbf{s}}{\epsilon}\right) + \alpha\mathbf{s} + \mathbf{C}(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \ddot{\mathbf{x}}_d - \hat{\mathbf{f}}\right) \tag{7.11}$$

where $\gamma > 0$ and $\alpha > 0$ are constant gains, and $\epsilon > 0$. To reduce the chattering in sliding mode control, the continuous function $\tanh(\cdot)$ is adopted to replace the discontinuous sign function [166]. ϵ determines the steepness of $\tanh(\cdot)$ as an approximation of the sign function. To prove the stability of the sliding mode control, we shall need the following lemma.

Lemma 7.2.1. *Let $f, V : [0 : \infty) \in R$, then $\dot{V} \leq -\alpha V + f$, $\forall t \geq t_0 \geq 0$ implies that [168],*

$$V(t) \leq e^{-\alpha(t-t_0)}V(t_0) + \int_{t_0}^{t} e^{-\alpha(t-\tau)} f(\tau) d\tau \tag{7.12}$$



Input
$\hat{\mathbf{f}}(\mathbf{x}, \dot{\mathbf{x}})$
$\hat{\mathbf{g}}(\mathbf{x}, \dot{\mathbf{x}})$
$\times\, \mathbf{u}$
$\hat{\ddot{\mathbf{x}}}$

Figure 7.5: A flowchart illustrating the neural networks model of a Delta robot.

Define a Lyapunov function as,

$$V = \frac{1}{2}\mathbf{s}^T\mathbf{s} \tag{7.13}$$

Then

$$\dot{V} = \mathbf{s}^T\dot{\mathbf{s}} \tag{7.14}$$

Since

$$\begin{aligned} \dot{\mathbf{s}} &= \mathbf{C}\dot{\mathbf{e}}(t) + \ddot{\mathbf{e}}(t) \\ &= \mathbf{C}(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + (\ddot{\mathbf{x}}_d - \ddot{\mathbf{x}}) \\ &= \mathbf{C}(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + (\ddot{\mathbf{x}}_d - \hat{\mathbf{f}} - \hat{\mathbf{g}}\mathbf{u}) \\ &= -\gamma \tanh\left(\frac{\mathbf{s}}{\epsilon}\right) - \alpha\mathbf{s} \end{aligned} \tag{7.15}$$

Therefore

$$\dot{V}(t) = \mathbf{s}^T \left(-\gamma \tanh\left(\frac{\mathbf{s}}{\epsilon}\right) - \alpha \mathbf{s}\right) \\ = -\alpha \mathbf{s}^T \mathbf{s} - \gamma \mathbf{s}^T \tanh\left(\frac{\mathbf{s}}{\epsilon}\right) \leq 0 \tag{7.16}$$

Hence, the closed-loop system is stable. Furthermore, we have

$$\dot{V}(t) = \begin{cases} -2(\alpha + \gamma/\epsilon)V(t) & s_i \leq \epsilon \text{ for all } i \\ -2\alpha V(t) & s_i > \epsilon \text{ for all } i \end{cases} \tag{7.17}$$

According to Lemma 7.2.1, we conclude that $V(t)$ exponentially decreases at the rate $2(\alpha + \gamma/\epsilon)$ when $\mathbf{s}$ is outside the boundary layer defined by ϵ, and at the rate 2α when $\mathbf{s}$ is inside the boundary layer. In both cases, we have

$$\lim_{t\to\infty} V(t) \to 0 \tag{7.18}$$

Hence, the tracking error of the sliding mode control for the system in Equation (7.8) converges asymptotically to zero at the exponential rate. When the neural networks model of the Delta robot has sufficiently small error, we would expect that the sliding mode control will deliver similar tracking performance in experiments. This is a subject of on-going work. The results will be reported in the future. In this paper, we shall apply the sliding mode control designed based on the neural networks model to the physics and geometry based model of the Delta robot to check the validity of the proposed modeling and control design approach.

In the following examples, we take $\mathbf{C}$ as $10 \times \mathbf{I}$ where $\mathbf{I}$ is the identity matrix, $\alpha = 0.1$, $\gamma = 0.1$, and $\epsilon = 1$.

Three different paths are adopted to test the performance of the proposed sliding mode control. The expressions of the three paths - heart curve, logarithmic spiral, and spiral are given as,

$$\text{Heart curve:} \begin{cases} x = 0.01(16 \sin^3(2.1\pi t/t_e)) \\ y = 0.01(13\cos(2.1\pi t/t_e) - 5\cos(4.1\pi t/t_e) \\ \qquad -2\cos(6.1\pi t/t_e) - \cos(8.1\pi t/t_e)) \\ z = -0.6 \end{cases} \tag{7.19}$$

$$\text{Logarithmic spiral:} \begin{cases} a = 0.2e^{(-0.48\pi t/t_e)} \\ x = a\cos(8\pi t/t_e) \\ y = -a\sin(8\pi t/t_e) \\ z = 0.1\sin(0.5\pi t/t_e) - 0.55 \end{cases} \tag{7.20}$$

$$\text{Spiral:} \begin{cases} x = 0.05\sin(5\pi t/t_e) \\ y = 0.22\cos(5\pi t/t_e) \\ z = -0.25t/t_e - 0.35 \end{cases} \tag{7.21}$$

where t_e is the end of the time. The 3D tracking results and part of the control inputs θ_i for different paths are shown from Figure 7.6 to Figure 7.11. The neural networks model-based sliding mode control demonstrates remarkable performance in trajectory tracking of various paths while maintaining bounded control inputs. This indicates that the trained neural networks model accurately captures the Delta robot's nonlinear property. The trajectory tracking errors of spiral curve for x, y, and z axis are shown in Figure 7.12 as an example. The errors of sliding mode control for different curves are shown in Figure 7.13.

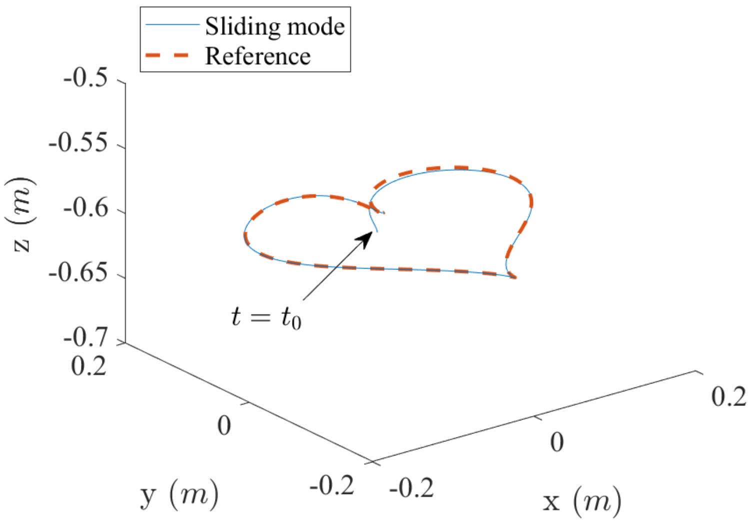


Figure 7.6: 3D trajectory tracking of heart curve using sliding mode control with neural networks model.

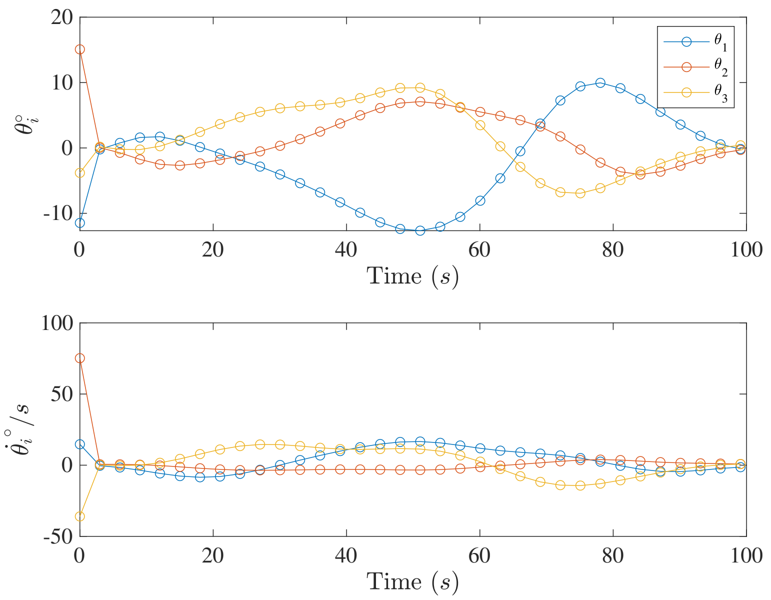


Figure 7.7: Control law from sliding mode control with neural networks model for heart curve trajectory tracking.

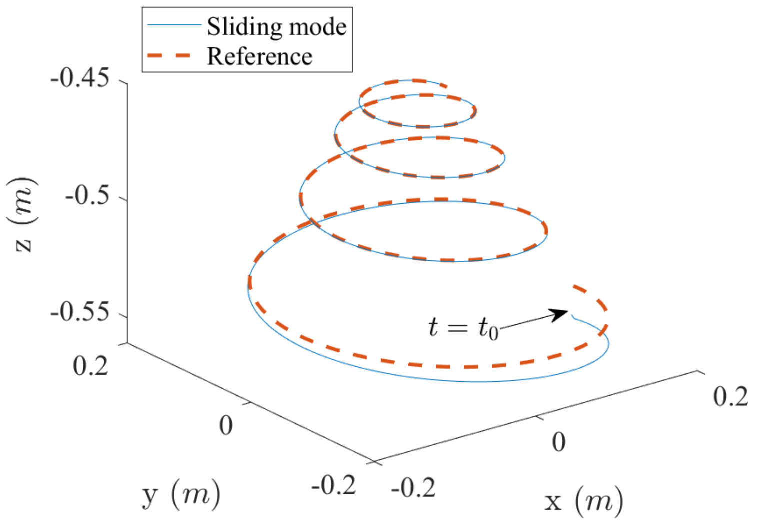


Figure 7.8: 3D trajectory tracking of logarithmic spiral curve using sliding mode control with neural network approximated model.

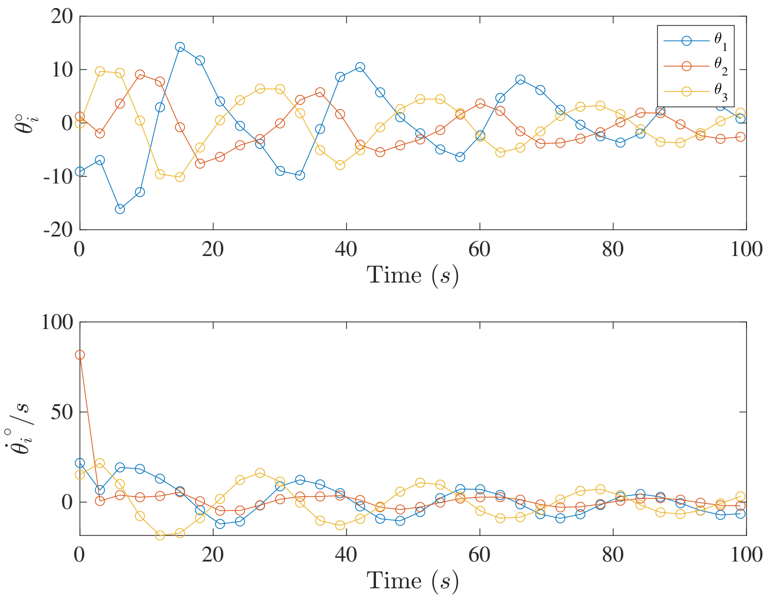


Figure 7.9: Control law from sliding mode control with neural networks model for logarithmic spiral curve trajectory tracking.

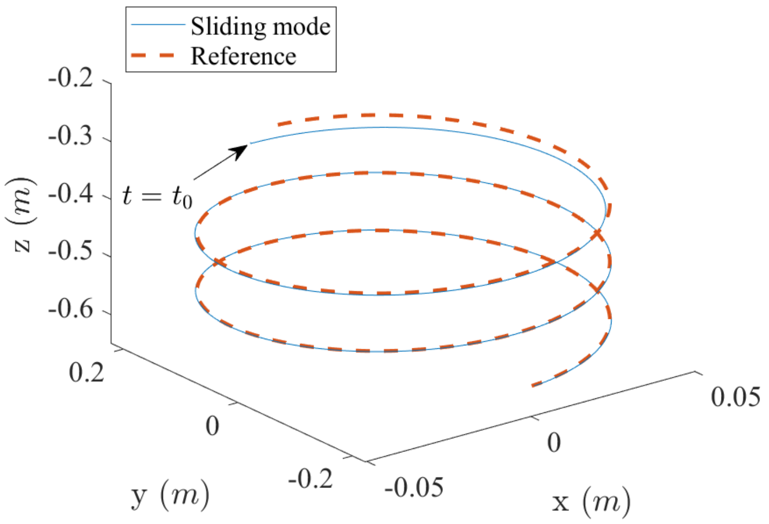


Figure 7.10: 3D trajectory tracking of spiral curve using sliding mode control with neural network model.

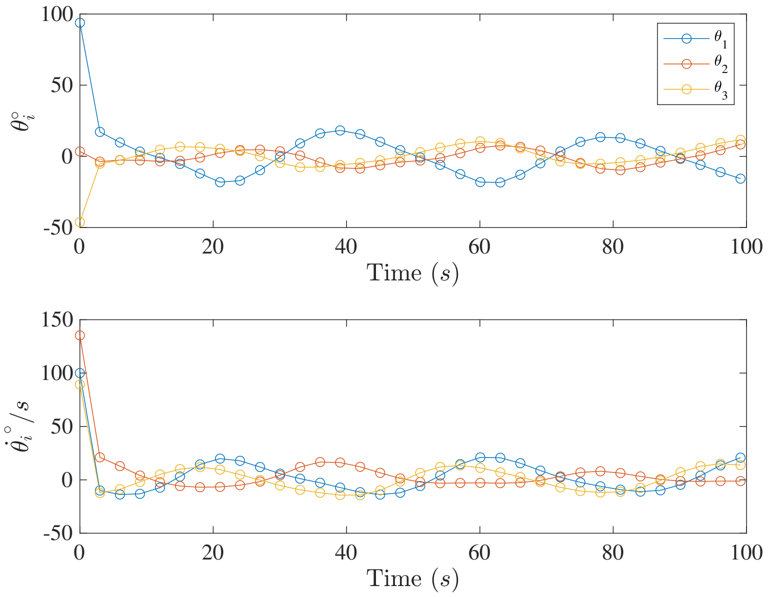


Figure 7.11: Control law from sliding mode control with neural networks model for spiral curve trajectory tracking.

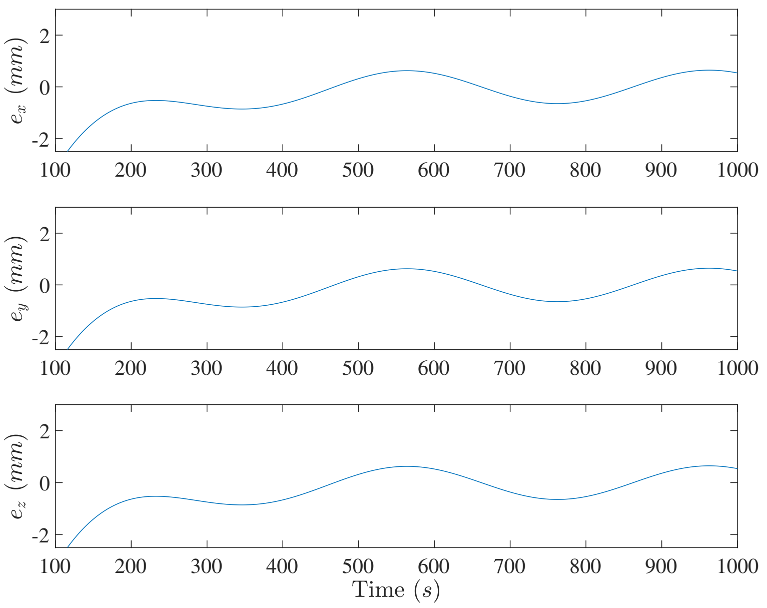


Figure 7.12: Spiral trajectory tracking errors for x, y, and z axis with sliding mode control.

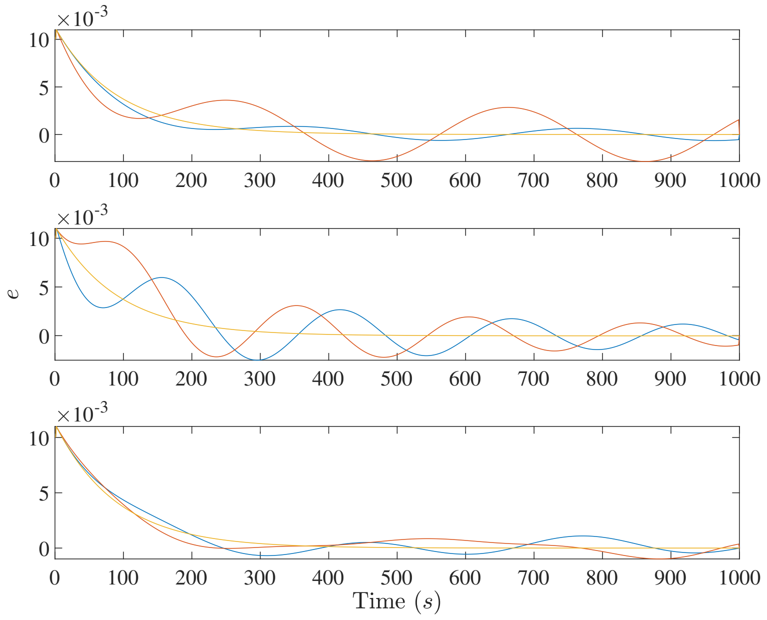


Figure 7.13: Error of sliding mode control in Equation (7.9) for trajectory tracking of different curves. Up: Error of sliding mode control for spiral trajectory tracking. Middle: Error of sliding mode control for logarithmic spiral trajectory tracking. Bottom: Error of sliding mode control for heart curve trajectory tracking.

7.3 Conclusions

Delta robot has highly nonlinear dynamics. Finding a control-affine model to describe the Delta robot analytically with joint angles and velocities as inputs is difficult. In this study, the neural networks are adopted to approximate the nonlinear model of the Delta robot with randomly sampled data in the workspace. Then, the sliding mode control is applied to the approximated model to track the desired trajectory. Extensive numerical results show that the neural networks model-based sliding mode control is highly effective for trajectory tracking for different paths. However, in real-world applications this control approach can be challenging due to differences in parameters of the frames, arms, and joints, as well as the presence of backlash and flexibility in the joints of the Delta robot. These factors can affect the implementation of neural networks based sliding mode control on hardware. Moreover, the parameters of the sliding mode control need to be fine-tuned based on the specific hardware being used, and high-frequency data collection is required for successful implementation. To overcome these challenges, further work is required to successfully implement neural networks-based sliding mode control on hardware.

Chapter 8

CONCLUSIONS AND FUTURE WORK

8.1 Concluding Remarks

This dissertation mainly explores model reduction and machine learning techniques to solve nonlinear optimal control problems. To improve the control perfomance and computational efficiency of optimal controls, various methods such as balanced truncation, empirical balanced truncation, radial basis neural networks, and transfer learning are adopted. Unlike the traditional LQR control approach, the proposed neural networks HJB solution solves the nonlinear problems and utilizes data to improve control performance. Except for solving the HJB equation, neural networks are also adopted to approximate the nonlinear functions for dynamic systems. The main focus of this dissertation is the neural networks solutions of optimal control. By leveraging both model and data, the performance of the optimal controller has been improved significantly. Combining data-driven algorithms and model-based control is a promising approach to enhancing optimal control performance. Moreover, the extended state observer and Luenberger observer are implemented to estimate the unknown dynamics or system states, facilitating the full-state feedback control design. Once the unknown dynamics are estimated, the recursive least squares algorithm is adopted to identify the unobservable or slow time-varying system parameters. The above mentioned algorithms have been tested on different engineering systems, including the high-speed train bogie, Quanser inverted pendulum, and Delta robot. Extensive numerical simulations and experimental results have shown that these algorithms are practical, efficient, and powerful for dynamic systems.

8.2 Future Work

8.2.1 RBFNN Optimal Control with Time-varying Dynamics

RBFNN have shown good performance in solving the optimal control problems for different dynamic systems. Nevertheless, it remains challenging to apply RBFNN when the system has complex dynamics, such as time-varying systems. In these situations, the value function $\mathbf{V}$ is a function of both state space and time, making the convergence of the solution more difficult. Consequently, it would be an interesting research topic to explore RBFNN optimal control for systems with time-varying dynamics for future work.

8.2.2 Optimization of RBFNN

In this dissertation, we adopted the RBFNN with fixed standard deviations and mean values to address the HJB equation solutions. Nonetheless, fixed standard deviations and mean values may not be optimal for different dynamic systems. To overcome this limitation, an optimization algorithm can be developed to solve the optimal control problem with trainable standard deviations, mean values, and weights. Several algorithms exist to optimize the RBFNN. However, exploring a more computationally efficient and generalized algorithm for future research will be very promising.

8.2.3 Hardware Implementation of Neural Networks based Sliding Mode Control

Numerically it shows that the neural networks model with sliding mode control has excellent trajectory tracking performance. However, in real-world applications the parameters of the frames, arms, and joints can be different. These factors can affect the implementation of neural networks based sliding mode control on hardware. The training of neural networks using experimental data is a highly challenging task. Moreover, the parameters of the sliding mode control need to be tuned based on the hardware. How to successfully implement the neural networks based sliding mode control on hardware can be a future work.

BIBLIOGRAPHY

[1] Quanser (2022) QUBE-Servo2. `https://www.quanser.com/products/qube-servo-2/`

[2] Yao Y, Wu G, Sardahi Y, Sun JQ (2018) Hunting stability analysis of high-speed train bogie under the frame lateral vibration active control. *Vehicle System Dynamics* **56**(2), 297–318

[3] Bryson Jr AE, Ho YC (1969) *Applied Optimal Control*. Blaisdell Publishing Company, Waltham, Massachusetts

[4] Vrabie D, Lewis F (2009) Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural Networks* **22**(3), 237–246

[5] Crandall MG, Lions PL (1983) Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society* **277**(1)

[6] Stefansson E, Leong YP (2016) Sequential alternating least squares for solving high dimensional linear Hamilton-Jacobi-Bellman equation. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, pp 3757–3764

[7] Greif C (2017) Numerical methods for Hamilton-Jacobi-Bellman equations. PhD thesis, University of Wisconsin-Milwaukee

[8] Lagaris IE, Likas AC, Papageorgiou DG (2000) Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks* **11**(5), 1041–1049

[9] Han J, Jentzen A, E W (2018) Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences* **115**(34), 8505–8510

[10] Sirignano J, Spiliopoulos K (2018) DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics* **375**, 1339–1364

[11] Weinan E, Yu B (2018) The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics & Statistics* **6**(1), 1–12

[12] Müller J, Zeinhofer M (2020) Deep Ritz revisited. *arXiv:191203937v2 [mathNA]*

[13] Weinan E, Han J, Jentzen A (2021) Algorithms for solving high dimensional PDEs: From nonlinear Monte Carlo to machine learning. *Nonlinearity* **35**(1), 278

[14] Munos R, Baird LC, Moore AW (1999) Gradient descent approaches to neural-net-based solutions of the Hamilton-Jacobi-Bellman equation. In: *Proceedings of International Joint Conference on Neural Networks*, vol 3, pp 2152–2157

[15] Nakamura-Zimmerer T, Gong Q, Kang W (2021) Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations. *SIAM Journal on Scientific Computing* **43**(2), A1221–A1247

[16] Abu-Khalaf M, Lewis FL (2005) Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica* **41**(5), 779–791

[17] Tassa Y, Erez T (2007) Least squares solutions of the HJB equation with neural network value-function approximators. *IEEE Transactions on Neural Networks* **18**(4), 1031–1041

[18] Liu D, Wang D, Wang FY, Li H, Yang X (2014) Neural-network-based online HJB solution for optimal robust guaranteed cost control of continuous-time uncertain nonlinear systems. *IEEE Transactions on Cybernetics* **44**(12), 2834–2847

[19] Jiang F, Chou G, Chen M, Tomlin CJ (2016) Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions. *arXiv preprint arXiv:161103158*

[20] Zhang D, Liu W, Qin C, Chen H (2016) Adaptive RBF neural-networks control for discrete nonlinear systems based on data. In: *Proceedings of the 12th World Congress on Intelligent Control and Automation*, IEEE, pp 2580–2585

[21] Yang X, Liu D, Wang D (2014) Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints. *International Journal of Control* **87**(3), 553–566

[22] Cheng T, Lewis FL, Abu-Khalaf M (2007) A neural network solution for fixed-final time optimal control of nonlinear systems. *Automatica* **43**(3), 482–490

[23] Cheng T, Lewis FL, Abu-Khalaf M (2007) Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach. *IEEE Transactions on Neural Networks* **18**(6), 1725–1737

[24] Lutter M, Belousov B, Listmann K, Clever D, Peters J (2020) HJB optimal feedback control with deep differential value functions and action constraints. In: *Conference on Robot Learning*, PMLR, pp 640–650

[25] Doya K (2000) Reinforcement learning in continuous time and space. *Neural Computation* **12**(1), 219–245

[26] Darbon J, Osher S (2016) Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere. *Research in the Mathematical Sciences* **3**(1), 1–26

[27] Yegorov I, Dower PM (2021) Perspectives on characteristics based curse-of-dimensionality-free numerical approaches for solving Hamilton–Jacobi equations. *Applied Mathematics & Optimization* **83**(1), 1–49

[28] Djeridane B, Lygeros J (2006) Neural approximation of PDE solutions: An application to reachability computations. In: *Proceedings of the 45th IEEE Conference on Decision and Control*, IEEE, pp 3034–3039

[29] Nakamura-Zimmerer T, Gong Q, Kang W (2020) A causality-free neural network method for high-dimensional Hamilton-Jacobi-Bellman equations. In: *2020 American Control Conference*, IEEE, pp 787–793

[30] Nakamura-Zimmerer TE (2022) A deep learning framework for optimal feedback control of high-dimensional nonlinear systems. Thesis, UC Santa Cruz

[31] Darbon J, Dower PM, Meng T (2021) Neural network architectures using min plus algebra for solving certain high dimensional optimal control problems and Hamilton-Jacobi PDEs. *arXiv preprint arXiv:210503336*

[32] Lewis FL, Vrabie D, Syrmos VL (2012) *Optimal control*. John Wiley & Sons

[33] Zhao A, Sun JQ (2022) Control for stability improvement of high-speed train bogie with a balanced truncation reduced order model. *Vehicle System Dynamics* pp 1–21

[34] Zhao A, Huang J, Sun JQ (2022) Estimation of wheel–rail structural interactions from motion signals of high-speed train bogie. *International Journal of Dynamics and Control* 10.1007/s40435-022-01085-2, `https://doi.org/10.1007/s40435-022-01085-2`

[35] Zhao A, Sun JQ (2022) Wear estimation of high speed train from motion measurements. *Journal of Vibration Testing and System Dynamics* **In press**

[36] Hespanha JP (2018) *Linear Systems Theory*. Princeton University Press

[37] Moore B (1981) Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control* **26**(1), 17–32

[38] Gugercin S, Antoulas AC (2004) A survey of model reduction by balanced truncation and some new results. *International Journal of Control* **77**(8), 748–766, 10.1080/00207170410001713448

[39] Pernebo L, Silverman L (1982) Model reduction via balanced state space representations. *IEEE Transactions on Automatic Control* **27**(2), 382–387

[40] Gawronski W, Juang JN (1990) Model reduction in limited time and frequency intervals. *International Journal of Systems Science* **21**(2), 349–376

[41] Laub A, Heath MT, Paige C, Ward R (1987) Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Transactions on Automatic Control* **32**(2), 115–122

[42] Hsu C, Hou D (1991) Reducing unstable linear control systems via real Schur transformation. *Electronics Letters* **27**(11), 984–986

[43] Bartels RH, Stewart GW (1972) Solution of the matrix equation $AX + XB = C$ [F4]. *Communications of the ACM* **15**(9), 820–826

[44] Gerrish F, Ward AJB (1998) Sylvester's matrix equation and Roth's removal rule. *The Mathematical Gazette* **82**(495), 423–430

[45] Van Der Schaft AJ (1992) l_2-gain analysis of nonlinear systems and nonlinear state feedback H_∞ control. *IEEE Transactions on Automatic Control* **37**(6), 770–784

[46] Phillips JR, Daniel L, Silveira LM (2003) Guaranteed passive balancing transformations for model order reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **22**(8), 1027–1041

[47] Obinata G, Anderson BDO (2012) *Model Reduction for Control System Design*. Springer Science & Business Media

[48] Besselink B, Tabak U, Lutowska A, van de Wouw N, Nijmeijer H, Rixen DJ, Hochstenbach ME, Schilders WHA (2013) A comparison of model reduction techniques from structural dynamics, numerical mathematics and systems and control. *Journal of Sound and Vibration* **332**(19), 4403–4422

[49] Beattie C, Gugercin S, Mehrmann V (2017) Model reduction for systems with inhomogeneous initial conditions. *Systems & Control Letters* **99**, 99–106

[50] Heinkenschloss M, Reis T, Antoulas AC (2011) Balanced truncation model reduction for systems with inhomogeneous initial conditions. *Automatica* **47**(3), 559–564

[51] Daraghmeh A, Hartmann C, Qatanani N (2019) Balanced model reduction of linear systems with nonzero initial conditions: Singular perturbation approximation. *Applied Mathematics and Computation* **353**, 295–307

[52] Cao X, Benner P, Pontes Duff I, Schilders W (2020) Model order reduction for bilinear control systems with inhomogeneous initial conditions. *International Journal of Control* pp 1–10

[53] Singh AK, Hahn J (2005) On the use of empirical gramians for controllability and observability analysis. In: *Proceedings of the 2005, American Control Conference, 2005.*, IEEE, pp 140–141

[54] Grundel S, Himpe C, Saak J (2019) On empirical system gramians. *PAMM* **19**(1), e201900,006

[55] Himpe C (2022) emgr–empirical gramian framework version 5.99. *arXiv preprint arXiv:220903833*

[56] Keil A, Gouzé JL (2003) Model reduction of modular systems using balancing methods. *Munich University of Technology Tech Rep*

[57] Csáji BC (2001) Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* **24**(48), 7

[58] Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. *Neural Computation* **3**(2), 246–257

[59] Medagam PV, Pourboghrat F (2009) Optimal control of nonlinear systems using RBF neural network and adaptive extended Kalman filter. In: *American Control Conference*, IEEE, pp 355–360

[60] Wu Y, Wang H, Zhang B, Du KL (2012) Using radial basis function networks for function approximation and classification. *ISRN Applied Mathematics* **2012**(doi:10.5402/2012/324194), 1–34

[61] Yu H, Xie T, Paszczyñski S, Wilamowski BM (2011) Advantages of radial basis function networks for dynamic system design. *IEEE Transactions on Industrial Electronics* **58**(12), 5438–5450

[62] Zhang W, Shen J, Ye X, Zhou S (2022) Error model-oriented vibration suppression control of free-floating space robot with flexible joints based on adaptive neural network. *Engineering Applications of Artificial Intelligence* **114**, 105,028

[63] Lewis F, Jagannathan S, Yesildirak A (1998) *Neural network control of robot manipulators and non-linear systems*. CRC press

[64] Sun JQ, Hsu CS (1988) A statistical study of generalized cell mapping. *Journal of Applied Mechanics* **55**, 694–701

[65] Silverman B (2018) *Density Estimation for Statistics and Data Analysis*. CRC Press, `https://books.google.com/books?id=fExnDwAAQBAJ`

[66] Chen JS, Hillman M, Chi SW (2017) Meshfree methods: Progress made after 20 years. *Engineering Mechanics* **143**, 04017,001, 10.1061/(ASCE)EM.1943-7889.0001176

[67] Wang X, Jiang J, Hong L, Sun JQ (2022) First-passage problem in random vibrations with radial basis function neural networks. *Journal of Vibration and Acoustics* **44**(October), 051,014–1–13

[68] Wang X, Jiang J, Hong L, Sun JQ (2022) Stochastic bifurcations and transient dynamics of probability responses with radial basis function neural networks. *International Journal of Non-Linear Mechanics* **147**, 104,244, https://doi.org/10.1016/j.ijnonlinmec.2022.104244

[69] Wang X, Jiang J, Hong L, Zhao A, Sun JQ (2023) Radial basis function neural networks solution for stationary probability density function of nonlinear stochastic systems. *Probabilistic Engineering Mechanics* **71**, 103,408, https://doi.org/10.1016/j.probengmech.2022.103408, `https://www.sciencedirect.com/science/article/pii/S0266892022001412`

[70] Saridis GN, Lee CSG (1979) An approximation theory of optimal control for trainable manipulators. *IEEE Transactions on Systems, Man, and Cybernetics* **9**(3), 152–159

[71] Meyn S (2022) *Control Systems and Reinforcement Learning*. Cambridge University Press

[72] Borovykh A, Kalise D, Laignelet A, Parpas P (2022) Data-driven initialization of deep learning solvers for Hamilton-Jacobi-Bellman PDEs. *arXiv preprint arXiv:220709299*

[73] Vamvoudakis KG, Lewis FL (2010) Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica* **46**(5), 878–888

[74] D Liu HL, Wang D (2014) Online synchronous approximate optimal learning algorithm for multi-player non-zero-sum games with unknown dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **44**(8), 1015–1027

[75] Modares H, Lewis FL (2014) Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning. *Automatica* **50**(7), 1780–1792

[76] H Modares FLL, Naghibi-Sistani MB (2014) Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica* **50**(1), 193–202

[77] B Luo TH H-N Wu, Liu D (2014) Data-based approximate policy iteration for affine nonlinear continuous-time optimal control design. *Automatica* **50**(12), 3281–3290

[78] Sundararajan N, Saratchandran P, Lu YW (1999) *Radial basis function neural networks with sequential learning: MRAN and its applications*, vol 11. World Scientific

[79] Kadirkamanathan V, Niranjan M (1993) A function estimation approach to sequential learning with neural networks. *Neural Computation* **5**(6), 954–975

[80] Platt J (1991) A resource-allocating network for function interpolation. *Neural Computation* **3**(2), 213–225

[81] Alwardi H, Wang S, Jennings LS, Richardson S (2012) An adaptive least-squares collocation radial basis function method for the HJB equation. *Journal of Global Optimization* **52**(2), 305–322

[82] Alwardi H, Wang S, Jennings LS (2013) An adaptive domain decomposition method for the Hamilton–Jacobi–Bellman equation. *Journal of Global Optimization* **56**(4), 1361–1373

[83] González-Casanova P, Muñoz-Gómez JA, Rodríguez-Gómez G (2009) Node adaptive domain decomposition method by radial basis functions. *Numerical Methods for Partial Differential Equations: An International Journal* **25**(6), 1482–1501

[84] Guo Y, Shi H, Kumar A, Grauman K, Rosing T, Feris R (2019) Spottune: transfer learning through adaptive fine-tuning. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 4805–4814

[85] Zabihifar SH, Yushchenko AS, Navvabi H (2020) Robust control based on adaptive neural network for rotary inverted pendulum with oscillation compensation. *Neural Computing and Applications* **32**(18), 14,667–14,679

[86] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. *arXiv preprint arXiv:150902971*

[87] Quanser (2021) Using the Reinforcement Learning Toolbox to balance an Inverted Pendulum. `https://www.quanser.com/blog/using-the-reinforcement-learning-toolbox-to-balance-an-inverted-pendulum/`

[88] Quanser (2022) Quanser QUBE-Servo2 pendulum control reinforcement learning. `https://www.mathworks.com/matlabcentral/fileexchange/106935-quanser-qube-servo-2-pendulum-control-reinforcement-learning`, [MATLAB Central File Exchange]

[89] Han J (2009) From PID to active disturbance rejection control. *IEEE Transactions on Industrial Electronics* **56**(3), 900–906

[90] Gao Z (2003) Scaling and bandwidth-parameterization based controller tuning. *Proceedings of the 2003 American Control Conference* **6**, 4989–4996

[91] Huang J, Lv Z, Sun JQ (2021) Optimal full-state feedback observer integrated backstepping control of chemical processes with unknown internal dynamics. *ISA Transactions*

[92] Yao J, Jiao Z, Ma D (2013) Adaptive robust control of DC motors with extended state observer. *IEEE Transactions on Industrial Electronics* **61**(7), 3630–3637

[93] Song Z, Xia C, Liu T (2012) Predictive current control of three-phase grid-connected converters with constant switching frequency for wind energy systems. *IEEE Transactions on Industrial Electronics* **60**(6), 2451–2464

[94] Zhang S, Wang Q, Dong C (2018) Extended state observer based control for generic hypersonic vehicles with nonaffine-in-control character. *ISA Transactions* **80**, 127–136

[95] Zhang X, Zhang X, Xue W, Xin B (2021) An overview on recent progress of extended state observers for uncertain systems: Methods, theory, and applications. *Advanced Control for Applications: Engineering and Industrial Systems* **3**(2), e89

[96] Zhang X, Xue W, Zhao Y (2021) On observability analysis and observer design for a class of nonlinear uncertain systems with general elastic vibration dynamics. *Asian Journal of Control*

[97] Hautus MLJ (1970) Stabilization controllability and observability of linear autonomous systems. *Indagationes Mathematicae (Proceedings)* **73**, 448–455

[98] Tian Y (1998) The Moore-Penrose inverses of $m \times n$ block matrices and their applications. *Linear Algebra and its Applications* **283**(1), 35–60

[99] Matsaglia G, PH Styan G (1974) Equalities and inequalities for ranks of matrices. *Linear and Multilinear Algebra* **2**(3), 269–292

[100] Åström KJ, Wittenmark B (2013) *Adaptive Control*. Courier Corporation, North Chelmsford, Massachusetts

[101] Zhai W, Liu P, Lin J, Wang K (2015) Experimental investigation on vibration behaviour of a CRH train at speed of 350 km/h. *International Journal of Rail Transportation* **3**(1), 1–16

[102] Yang CD, Sun YP (2001) Mixed H_2/H_∞ cruise controller design for high speed train. *International Journal of Control* **74**(9), 905–920

[103] Dong H, Ning B, Cai B, Hou Z (2010) Automatic train control system development and simulation for high-speed railways. *IEEE Circuits and Systems Magazine* **10**(2), 6–18

[104] Perez J, Busturia JM, Goodall RM (2002) Control strategies for active steering of bogie-based railway vehicles. *Control Engineering Practice* **10**(9), 1005–1012

[105] Mei TX, Goodall RM (2003) Recent development in active steering of railway vehicles. *Vehicle System Dynamics* **39**(6), 415–436

[106] Abood KHA, Khan RA (2010) Investigation to improve hunting stability of railway carriage using semi-active longitudinal primary stiffness suspension. *Journal of Mechanical Engineering Research* **2**(5), 97–105

[107] Zong LH, Gong XL, Xuan SH, Guo CY (2013) Semi-active H_∞ control of high-speed railway vehicle suspension with magnetorheological dampers. *Vehicle System Dynamics* **51**(5), 600–626

[108] Sharma SK, Kumar A (2018) Disturbance rejection and force-tracking controller of nonlinear lateral vibrations in passenger rail vehicle using magnetorheological fluid damper. *Journal of Intelligent Material Systems and Structures* **29**(2), 279–297

[109] Pearson JT, Goodall RM, Mei TX, Himmelstein G (2004) Active stability control strategies for a high speed bogie. *Control Engineering Practice* **12**(11), 1381–1391

[110] Yao Y, Li G, Wu G, Zhang Z, Tang J (2020) Suspension parameters optimum of high-speed train bogie for hunting stability robustness. *International Journal of Rail Transportation* **8**(3), 195–214

[111] Huang Y, Kramer B (2020) Balanced reduced-order models for iterative nonlinear control of large-scale systems. *IEEE Control Systems Letters* **5**(5), 1699–1704

[112] Wu Y, Hamroun B, Le Gorrec Y, Maschke B (2020) Reduced order LQG control design for infinite dimensional port Hamiltonian systems. *IEEE Transactions on Automatic Control*

[113] Zolotas AC, Pearson JT, Goodall R (2006) Modelling requirements for the design of active stability control strategies for a high speed bogie. *Multibody System Dynamics* **15**(1), 51–66

[114] Uhl T (2007) The inverse identification problem and its technical application. *Archive of Applied Mechanics* **77**(5), 325–337

[115] Xia F, Cole C, Wolfs P (2007) An inverse railway wagon model and its applications. *Vehicle System Dynamics* **45**(6), 583–605

[116] Xia F, Cole C, Wolfs P (2008) Grey box-based inverse wagon model to predict wheel–rail contact forces from measured wagon body responses. *Vehicle System Dynamics* **46**(S1), 469–479

[117] Wei L, Zeng J, Wu P, Gao H (2014) Indirect method for wheel–rail force measurement and derailment evaluation. *Vehicle System Dynamics* **52**(12), 1622–1641

[118] Zeng J, Wei L, Wu P (2016) Safety evaluation for railway vehicles using an improved indirect measurement method of wheel–rail forces. *Journal of Modern Transportation* **24**(2), 114–123

[119] Charles G, Goodall R, Dixon R (2008) Model-based condition monitoring at the wheel–rail interface. *Vehicle System Dynamics* **46**(S1), 415–430

[120] Strano S, Terzo M (2018) On the real-time estimation of the wheel-rail contact force by means of a new nonlinear estimator design model. *Mechanical Systems and Signal Processing* **105**, 391–403

[121] Muñoz S, Ros J, Urda P, Escalona JL (2021) Estimation of lateral track irregularity through Kalman filtering techniques. *IEEE Access* **9**, 60,010–60,025

[122] Ward CP, Goodall RM, Dixon R, Charles G (2012) Adhesion estimation at the wheel–rail interface using advanced model-based filtering. *Vehicle System Dynamics* **50**(12), 1797–1816

[123] Onat A, Voltr P, Lata M (2017) A new friction condition identification approach for wheel–rail interface. *International Journal of Rail Transportation* **5**(3), 127–144

[124] Mal K, Hussain I, Chowdhry BS, Memon TD (2020) Extended Kalman filter for estimation of contact forces at wheel-rail interface. *3C Tecnología*

[125] Hui J, Yuan J (2022) Kalman filter, particle filter, and extended state observer for linear state estimation under perturbation (or noise) of MHTGR. *Progress in Nuclear Energy* **148**, 104,231

[126] Gadhave R, Vyas N (2021) Rail-wheel contact forces and track irregularity estimation from on-board accelerometer data. *Vehicle System Dynamics* pp 1–22

[127] Nefti S, Oussalah M (2004) A neural network approach for railway safety prediction. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, IEEE, vol 4, pp 3915–3920

[128] Shen Z, Hedrick J, Elkins J (1983) A comparison of alternative creep force models for rail vehicle dynamic analysis. *Vehicle System Dynamics* **12**(1-3), 79–83

[129] Lin B, Wu J, Lin R, Wang J, Wang H, Zhang X (2019) Optimization of high-level preventive maintenance scheduling for high-speed trains. *Reliability Engineering & System Safety* **183**, 261–275

[130] Kalker JJ (1979) Survey of wheel-rail rolling contact theory. *Vehicle System Dynamics* **8**(4), 317–358

[131] Kalker JJ (1991) Wheel-rail rolling contact theory. *Wear* **144**(1-2), 243–261

[132] Knothe K (2008) History of wheel/rail contact mechanics: from Redtenbacher to Kalker. *Vehicle System Dynamics* **46**(1-2), 9–26

[133] Archard J (1953) Contact and rubbing of flat surfaces. *Journal of Applied Physics* **24**(8), 981–988

[134] Zobory I (1997) Prediction of wheel/rail profile wear. *Vehicle System Dynamics* **28**(2-3), 221–259

[135] Myśliński A, Chudzikiewicz A (2021) Wear modelling in wheel–rail contact problems based on energy dissipation. *Tribology-Materials, Surfaces & Interfaces* **15**(2), 138–149

[136] Jendel T (2002) Prediction of wheel profile wear – comparisons with field measurements. *Wear* **253**(1-2), 89–99

[137] Jin XC (2012) Wheel wear predictions and analyses of high-speed trains. *Nonlinear Engineering* **1**(3-4), 91–100

[138] Li Y, Ren Z, Enblom R, Stichel S, Li G (2020) Wheel wear prediction on a high-speed train in China. *Vehicle System Dynamics* **58**(12), 1839–1858

[139] Li X, Ding Q, Sun JQ (2018) Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety* **172**, 1–11

[140] Özel T, Karpat Y (2005) Predictive modeling of surface roughness and tool wear in hard turning using regression and neural networks. *International Journal of Machine Tools and Manufacture* **45**(4-5), 467–479

[141] Shebani A, Iwnicki S (2018) Prediction of wheel and rail wear under different contact conditions using artificial neural networks. *Wear* **406**, 173–184

[142] Wang S, Yan H, Liu C, Fan N, Liu X, Wang C (2021) Analysis and prediction of high-speed train wheel wear based on SIMPACK and backpropagation neural networks. *Expert Systems* **38**(7), e12,417

[143] Weinstock H (1984) Wheel climb derailment criteria for evaluation of rail vehicle safety. In: *Proceedings of ASME Winter Annual Meeting*, New Orleans, Louisiana

[144] Shi X, Yan Q, Zhang X, Diao G, Zhang C, Hong Z, Wen Z, Jin X (2019) Hardness matching of rail/wheel steels for high-speed-train based on wear rate and rolling contact fatigue performance. *Materials Research Express* **6**(6), 066,501

[145] Zhao H, Liu P, Ding Y, Jiang B, Liu X, Zhang M, Chen G (2020) An investigation on wear behavior of ER8 and SSW-Q3R wheel steel under pure rolling condition. *Metals* **10**(4), 513

[146] Garg V (2012) *Dynamics of Railway Vehicle Systems*. Elsevier

[147] Ayasse JB, Chollet H (2006) Wheel-rail contact. *Handbook of Railway Vehicle Dynamics* pp 85–120

[148] Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. `arXiv: 1412.6980`

[149] Feuersänger C (2011) Manual for package pgfplots. `http://www.ctan.org/tex-archive/help/Catalogue/entries/pgfplots.html`

[150] Castañeda LA, Luviano-Juárez A, Chairez I (2014) Robust trajectory tracking of a Delta robot through adaptive active disturbance rejection control. *IEEE Transactions on control systems technology* **23**(4), 1387–1398

[151] Angel L, Viola J (2018) Fractional order PID for tracking control of a parallel robotic manipulator type delta. *ISA transactions* **79**, 172–188

[152] Kuo YL, Huang PY (2017) Experimental and simulation studies of motion control of a Delta robot using a model-based approach. *International Journal of Advanced Robotic Systems* **14**(6), 1729881417738,738

[153] López M, Castillo E, García G, Bashir A (2006) Delta robot: inverse, direct, and intermediate jacobians. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* **220**(1), 103–109

[154] Zubizarreta A, Larrea M, Irigoyen E, Cabanes I, Portillo E (2018) Real time direct kinematic problem computation of the 3PRS robot using neural networks. *Neurocomputing* **271**, 104–114

[155] Abed Azad F, Ansari Rad S, Hairi Yazdi MR, Tale Masouleh M, Kalhor A (2022) Dynamics analysis, offline–online tuning and identification of base inertia parameters for the 3-DOF Delta parallel robot under insufficient excitations. *Meccanica* **57**(2), 473–506

[156] Xu J, Wang Q, Lin Q (2018) Parallel robot with fuzzy neural network sliding mode control. *Advances in Mechanical Engineering* **10**(10), 1687814018801,261

[157] Boudjedir CE, Boukhetala D, Bouri M (2018) Nonlinear PD plus sliding mode control with application to a parallel Delta robot. *Journal of Electrical Engineering-Elektrotechnicky Casopis* **69**(ARTICLE), 329–336

[158] Pham PC, Kuo YL (2022) Robust adaptive finite-time synergetic tracking control of Delta robot based on radial basis function neural networks. *Applied Sciences* **12**(21), 10,861

[159] Zhao R, Wu L, Chen YH (2020) Robust control for nonlinear Delta parallel robot with uncertainty: an online estimation approach. *IEEE Access* **8**, 97,604–97,617

[160] Boudjedir CE, Bouri M, Boukhetala D (2019) Iterative learning control for trajectory tracking of a parallel Delta robot. *at-Automatisierungstechnik* **67**(2), 145–156

[161] Boudjedir CE, Bouri M, Boukhetala D (2020) Model-free iterative learning control with nonrepetitive trajectories for second-order MIMO nonlinear systems—application to a Delta robot. *IEEE Transactions on Industrial Electronics* **68**(8), 7433–7443

[162] Ghafarian Tamizi M, Ahmadi Kashani AA, Abed Azad F, Kalhor A, Masouleh MT (2022) Experimental study on a novel simultaneous control and identification of a 3-DOF Delta robot using model reference adaptive control. *European Journal of Control* **67**, 100,715

[163] Mueller A (2019) Modern robotics: Mechanics, planning, and control. *IEEE Control Systems Magazine* **39**(6), 100–102

[164] Bottou L (2012) Stochastic gradient descent tricks. *Neural Networks: Tricks of the Trade: Second Edition* pp 421–436

[165] Liu J (2017) *Sliding mode control using MATLAB*. Academic Press

[166] Aghababa MP, Akbari ME (2012) A chattering-free robust adaptive sliding mode controller for synchronization of two different chaotic systems with unknown uncertainties and external disturbances. *Applied Mathematics and Computation* **218**(9), 5757–5768

[167] Polycarpou MM, Ioannou PA (1993) A robust adaptive nonlinear control design. In: *1993 American control conference*, IEEE, pp 1365–1369

[168] Ioannou PA, Sun J (1996) *Robust adaptive control*, vol 1. PTR Prentice-Hall Upper Saddle River, NJ

Chapter 9
APPENDIX

9.1 Bogie Model

The matrices $\mathbf{M}$, $\mathbf{C}_D$, $\mathbf{K}$ and $\mathbf{E}$ of the bogie are given as [2],

$$\mathbf{M} = \begin{bmatrix} m_w &&&&&&& \\ & I_w &&&&&& \\ && m_w &&&&& \\ &&& I_w &&&& \\ &&&& m_f &&& \\ &&&&& I_f && \\ &&&&&& m_m & \\ &&&&&&& m_m \end{bmatrix}$$

$$\mathbf{C}_D = \begin{bmatrix} \frac{2f_\eta}{v} &&&&&&& \\ & \frac{2l_0^2 f_\xi}{v} &&&&&& \\ && \frac{2f_\eta}{v} &&&&& \\ &&& \frac{2l_0^2 f_\xi}{v} &&&& \\ &&&& c_{sy}+2c_{my} && -c_{my} & -c_{my} \\ &&&&& l_2^2 c_{sx} + l_m^2 c_{my} & -l_m c_{my} & l_m c_{my} \\ &&&& -c_{my} & -l_m c_{my} & c_{my} & \\ &&&& -c_{my} & l_m c_{my} && c_{my} \end{bmatrix}$$

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & l_e & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -l_e & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix}$$

where

$$\mathbf{K}_{11} = \begin{bmatrix} k_{py} & & & \\ & l_1^2 k_{px} & & \\ & & k_{py} & \\ & & & l_1^2 k_{px} \end{bmatrix}$$

$$\mathbf{K}_{12} = \mathbf{K}_{21} = \begin{bmatrix} -k_{py} & -bk_{py} & 0 & 0 \\ 0 & -l_1^2 k_{px} & 0 & 0 \\ -k_{py} & bk_{py} & 0 & 0 \\ 0 & l_1^2 k_{px} & 0 & 0 \end{bmatrix}$$

$$\mathbf{K}_{22} = \begin{bmatrix} 2k_{py} + k_{sy} + 2k_{my} & 0 & -k_{my} & -k_{my} \\ 0 & k_{66} & -l_m k_{my} & l_m k_{my} \\ -k_{my} & -l_m k_{my} & k_{my} & 0 \\ -k_{my} & l_m k_{my} & 0 & k_{my} \end{bmatrix}$$

and $k_{66} = l_2^2 k_{sx} + 2l_1^2 k_{px} + 2b^2 k_{py} + 2l_m^2 k_{my}$.

Table 9.1: Parameters of the bogie system.

Symbol	Description	Value
m_w	Mass of the wheelset	2585 kg
I_w	Yaw inertia of the wheelset	2024 kg·m^2
m_f	Mass of the frame	7186 kg
I_f	Yaw inertia of the frame	9.75×10^3 m
m_m	Mass of motor	1765 kg
λ_e	Wheel-rail contact conicity	0.1
k_{px}	Primary longitudinal stiffness per axle	40 kN/mm
k_{py}	Primary lateral stiffness per axle	6 kN/mm
k_{sx}	Secondary longitudinal stiffness	0.8 kN/mm
k_{sy}	Secondary lateral stiffness	0.8 kN/mm
c_{sx}	Yaw damper damping	10 kN·s/m
c_{sy}	Secondary lateral stiffness	60 kN·s/m
$2l_1$	Lateral spacing of primary suspension	2.2 m
l_e	Longitudinal distance from end beam	1 m
$2b$	Wheel base	2.5 m
$2l_0$	Distance of the contact spot	1.493 m
$2l_2$	Lateral spacing of the secondary suspension	1.9 m
l_m	Distance from motor suspension to the frame	0.5 m
r_0	Wheel rolling radius	0.625 m
f_ζ	Longitudinal creep coefficient	8.144×10^6 N
f_η	Lateral creep coefficient	8.624×10^6 N
f_{my}	Motor suspension frequency	1.8 Hz
ξ_{my}	Motor suspension damping ratio	0.2
k_{gy}	Gravitational stiffness	-
$k_{g\psi}$	Secondary lateral stiffness	-
v	Train speed	360 km/h